WebWork2文档中文化计划 : WebWork Wiki Chinese Version 1.0a

This page last changed on May 26, 2006 by scud.

版本:1.0a -->开始阅读

WebWork文档中文化小组 2006-5-27

今天你为开源做了什么?

在线文档/下载

- 文档下载: http://webwork.javascud.org (Html和Pdf格式)
- 在线文档: http://wiki.javascud.org/display/ww2cndoc/
- 文档目录: http://wiki.javascud.org/display/ww2cndoc/WebWork
- 英文文档: http://wiki.opensymphony.com/display/WW/WebWork

文档中文化计划发起网站

- WebWork China
- JavaScud 开源平台

版权声明

WebWork2文档中文化小组成员

(按报名顺序排列)

网名	个人说明	个人网站/博客	赠言
scud(飞云小侠)	WebWork China 发起人 JavaScud开源平台发起人	飞云小侠的博客	开源需要敬业和坚持 帮助别人就是帮助自己
zjumty(黑灵)	一名文科出身普通的开发人员	The Developer Side	喜欢编程序, 愿结识更多的业内人士
Tin(Tin Steeler)	Quake3玩家出身的 Web应用开发人员	Tin's Blog	习惯改变, 修身养性
Foxcoming(Jeff Wang)	愉快的程序员		Open the Source of Innovation, it's "Open Source"!
yangkaifeng	"懒人"一个		以更懒的方法更轻松的干好活

darren.hoo			
kaktos	没头脑且不高兴的完美主义者	Beta Life	Ignorance is a bliss
losingfox(Leo Sun)	乐于探索的J2EE开发者		work smart and enjoy your life

文档内容简介

按照WebWork 2的Wiki,本小组对WebWork的文档进行了翻译.目前最重要的部分"Reference",除部分过时文档外已经全部翻译完毕.其他部分也翻译了一部分.在后续的工作中,我们将继续进行翻译其它部分.

已经翻译的主要有:

- Reference
  - Basics
  - Advaneced
  - UI/Views
  - J2SE5 Support
  - Portlet

其他部分(不是全部):

- About WebWork
- Getting Started
- 3rd Party Integration
  - Spring
  - SiteMesh
- Previous releases

其他事宜

- 由于小组成员水平有限,时间紧迫,文档中肯定存在翻译错误,遗漏,不通顺等问题,请在相关文档的页面发表备注,指出我们的错误,方便修正.方便大家,方便自己.
- 文档内标注了"译注"的内容,均由翻译人员添加.

WebWork2文档中文化计划 : 阅读指南

This page last changed on May 26, 2006 by scud.

文档格式

目前发布的文档格式包含PDF格式和HTML格式两种.

阅读 Html 文档

下载文档后点击index.htm可以阅读,在每页具有返回目录的链接.

阅读 PDF 文档

PDF文档由于页面顺序是不按照实际顺序的,所以阅读PDF时请按照文档目录或者PDF左侧书签的顺序阅读.
(Confluence目前不支持定义页面顺序,手动调整工作量太大)

有几个PDF表格内的内容超出表格范围,应该是程序错误造成的.建议浏览对应的HTML版本或者在线浏览.

过时文档

除"Reference"部分,其他部分有很多页面都已经是过时文档了,阅读时请注意区分,以免混淆.

WebWork

This page last changed on May 22, 2006 by tin.

欢迎来到 **WebWork** 的wiki. WebWork的官方网址为 http://www.opensymphony.com/webwork/. 在那里你可以找到最新发布版本的WebWork的文档. 这个 wiki 是用来编写一些补充的信息同时也是最新的开发版本的文档.

关于WebWork

- WebWork概述
- 发行公告
- 历史, 开发团队
- 贡献(Contributing)
- 错误与问题跟踪 (直接报告)
- 文章, 发布通讯稿, 用户评价, 使用WebWork的项目
- 与其它Web框架比较
- 许可(License)
- 关于Struts Action 2.0的信息

入门指南

- 下载WebWork (continous builds, previous versions)
- 安装
- 运行要求 (依赖, 应用服务器)
- 快速上手
- 教程
- 构建WebWork

参考手册

- API
- 基础
  - 体系结构
  - 配置文件 (web.xml, xwork.xml, webwork.properties, webwork\-default.xml, velocity.properties)
  - XWork配置
  - 拦截器
  - Result 类型
  - 异常处理
- 高级功能
  - Action链
  - 控制反转(IoC)
  - 类型转换
  - 校验
  - 国际化
  - Continuations
  - ActionMapper
- UI/视图
  - 标签(Tags)
    - 通用标签 (控制标签, 数据标签)
    - HTML标签 (表单标签, 非表单标签)
  - OGNL (syntax, altsyntax)
  - 视图技术
    - JSP (相关的标签)
    - Freemarker (相关的标签)
    - Velocity (相关的标签)
    - JasperReports
  - 模板和主题

    - 组件
- J2SE 5 支持 (标注)
- Portlets (教程, 配置)

用户(Users)

- 测试
- 代码片断 和 示例应用程序

开发人员& 贡献者(Contributor)

- 文档编写风格指导
- 源代码
- Java.net项目主页
- Ivy

集成,工具,IDE

- IDE 插件: Eclipse, IntelliJ
- 有关工具 (SiteGraph, Config Browser, QuickStart)
- 与第三方集成: SiteMesh, Spring, Pico, Hibernate, JSTL, JUnit, Quartz, Tiles

帮助

- 常见问题解答(FAQ) & The Vault
- 邮件列表
- 论坛
    - WebWork用户论坛 (direct)
    - WebWork开发者论坛 (direct)
- Chat (聊天记录)

增补

译注
此处为翻译时加上的,原文不包含此部分目录,但是内容都是原来有的,这样可以方便列出一些没有包含在任何页面的节点

- 关于WebWork
- 以前的发行公告
- WebWork 2.2迁移事项
- Misc

WebWork Overview

This page last changed on May 19, 2006 by scud.

WebWork是一个强大的基于Web的MVC框架, 它构建在一个command模式的框架 XWorkXWork 之上. WebWork真正的力量在于它内在蕴涵的简单和协作的理念. 使用WebWork将有助于最小化代码,并允许开发人员更多的关注业务逻辑(business logic)和建模(modeling), 而不是关注构建基于web的应用程序必需的铅管系统(plumbing).

特性

- 灵活的校验Validation框架,允许我们的校验规则与action代码解耦(decoupled).
- 类型转化Type Conversion 允许我们很容易的将对象从一个类转换成另一个类, 解除创建web应用时最乏味单调的工作之一.
- 强大的*表达式语言(Expression Language)* 基于OGNL之上, 允许遍历动态对象图, 执行对象方法以及使用值栈(ValueStack)实现对多个JavaBean属性的透明访问. Webwork也可以使用JSTL.
- 控制反转 集成,管理组件的生命周期和依赖,不需要client必须进行调用来获取一个组件实例的方式来注册类.WebWork推荐使用Spring进行IoC.
- 可复用的 Tags 允许使用Themes and Templates 进行轻松的和可复用的组件化web开发.
- 高级的 Interceptors 提供了各种丰富的功能,包括防止多次表单提交,在后台执行长时间的查询.
- 对国际化的多层次和可插入的支持.
- 轻松地和第三方软件结合,包括 Hibernate, Spring, Sitemesh, 和 JSTL.
- 支持多种视图技术,例如 JSP, FreeMarker 和 JasperReports.
- 模块化的 配置文件 ,使用包(package)和命名空间(namespace)来管理众多的action.
- 通过ajax theme提供了高级的AJAX 特性.

Release Notes

This page last changed on May 22, 2006 by tin.

WebWork 2.2.2 发行公告

主要变化

Portlets

- Portlet 增加了测试, logging 和 url 已经修正
- 增加了对Velocity的支持

校验

- 改进了 客户端校验

UI 和视图

- 新的和改进的组件: Submit 现在支持图像和html按钮, Reset, RichTextEditor, url, Date, Token
- 各种组件都有了tooltip(提示)支持
- Freemarker 和 Velocity 的bug修正
- Freemarker 和 JSTL更好地和SiteMesh结合
- 很多xhtml和ajax theme的错误修正
- 增加对使用Tiles的支持
- 支持新的普通文本 Result类型

工具

- Quickstart OSX 问题修正

其他

- 更新和改进了文档,包括新的 开始启动 页面和 Portlet 教程
- 更新了测试覆盖率

迁移注意事项

WebWork 2.2.2 是一个错误修正发布版本,也包含一些小的改进,大多数和视图层相关. 迁移到 2.2.2 可能是很简单的,需要注意一个Param 标签.
WebWork 2.2.2 是在Opensymphony光环下的最后一次发布.WebWork 现在已经移动到Apache组织,将会作为计划在2006年8月发布的新的Struts Action 2.0的基础.

修改记录

要查看完整的修改记录,请查看 complete changelog

已知问题

XSL Result 在Java 5.0上存在问题.

OpenSymphony JIRA (113 issues)

T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
●	WW-1265	DatePicker - locale/language problems	Rene Gielen	Claus Ibsen	↑	Closed	FIXED	Mar 23, 2006	Mar 24, 2006	
●	WW-1264	Fix broken calender i18n js scripts	Rene Gielen	Rene Gielen	↑	Closed	FIXED	Mar 23, 2006	Mar 23, 2006	
●	WW-1263	config-browser - can not list interceptors correctly	Rainer Hermanns	Claus Ibsen	↑	Closed	FIXED	Mar 23, 2006	Mar 23, 2006	
●	WW-1262	showcase - some issues	Rene Gielen	Claus Ibsen	↓	Closed	FIXED	Mar 23, 2006	Mar 23, 2006	
↗	WW-1261	add a test directory and a sample test case to webapp "ant new" target	tm_jee	tm_jee	↑	Resolved	FIXED	Mar 22, 2006	Mar 22, 2006	
↗	WW-1258	Ignoring ClientDisconnectExceptions in the log file	Rene Gielen	Jeroen van Vianen	↓	Closed	WON'T FIX	Mar 21, 2006	Mar 21, 2006	
+	WW-1256	Create a ww:reset component to work along with ww:submit	Rene Gielen	Rene Gielen	↑	Closed	FIXED	Mar 20, 2006	Mar 21, 2006	
▣	WW-1254	zipped distribution should contain a project directory webwork-X.Y.Z	Alexandru Popescu	Rainer Hermanns	↑	Resolved	FIXED	Mar 20, 2006	Mar 20, 2006	
+	WW-1253	Default Action Support	Unassigned	Andres March	↓	Resolved	FIXED	Mar 19, 2006	Mar 19, 2006	
●	WW-1251	Running ant new in webapps fails	Claus Ibsen	Eric Molitor	↑	Closed	FIXED	Mar 19, 2006	Mar 22, 2006	
●	WW-1249	FileUploadInterceptor not using internationalized error message	Don Brown	tm_jee	↑	Resolved	FIXED	Mar 18, 2006	Mar 18, 2006	

●	WW-1248	DatePicker single click not working	Rene Gielen	Claus Ibsen	⬇	✓ Closed	FIXED	Mar 17, 2006	Mar 18, 2006
▣	WW-1247	Document interceptor properties overriding on action configuration level	tm_jee	Rene Gielen	⬆	✓ Closed	FIXED	Mar 17, 2006	Mar 18, 2006
▣	WW-1246	Document excludeMethods / includeMethods parameters for validator, token and workflow interceptor	tm_jee	Rene Gielen	⬆	✓ Closed	FIXED	Mar 17, 2006	Mar 18, 2006
↗	WW-1245	Add an example for the Date component in showcase app	Rainer Hermanns	Rene Gielen	⬆	✓ Closed	FIXED	Mar 17, 2006	Mar 21, 2006
●	WW-1244	Velocity doubleselect tag exmple does not work as expected in showcase application	Rene Gielen	Rene Gielen	⬆	✓ Closed	FIXED	Mar 17, 2006	Mar 18, 2006
●	WW-1243	doubleselect tag causes exception, indicating invalid stack constitution assumption	Rene Gielen	Rene Gielen	⬇	➡ Resolved	FIXED	Mar 17, 2006	Mar 18, 2006
●	WW-1242	action attribute in form ads .action onto value	Rainer Hermanns	Ian Roughley	⬆	✓ Closed	FIXED	Mar 17, 2006	Mar 17, 2006
↗	WW-1241	Have ObjectFactoryDestroyable and ObjectFactoryLifecycle (in addition to the current ObjectFactoryInitlalizable)	tm_jee	tm_jee	⬆	➡ Resolved	FIXED	Mar 16, 2006	Mar 16, 2006

↗	WW-1240	Improve component parameter documentation to have real javadoc	Rene Gielen	Rene Gielen	⇧	Closed	FIXED	Mar 16, 2006	Mar 19, 2006
●	WW-1239	MultiPartRequestWrapper uses Class.forName resulting in 500 error	Rainer Hermanns	Nick Hill	⇧	Resolved	FIXED	Mar 15, 2006	Mar 16, 2006
↗	WW-1238	Fix Getting Started Documentation	Rene Gielen	Rene Gielen	⇧	Resolved	FIXED	Mar 14, 2006	Mar 23, 2006
↗	WW-1237	Refine logging	Nils-Helge Garli	Nils-Helge Garli	⇩	Closed	FIXED	Mar 13, 2006	Mar 17, 2006
●	WW-1236	Missing test for null &apos;action&apos; property in start() of URL component?	Unassigned	Nils-Helge Garli	⇧	Resolved	FIXED	Mar 13, 2006	Mar 25, 2006
↗	WW-1235	JakartaMultiPartRequest - refactor to not used deprecated methods	Claus Ibsen	Claus Ibsen	⇩	Closed	FIXED	Mar 11, 2006	Mar 11, 2006
↗	WW-1234	replace deprecated way of geting Freemarker&apos;s Configuration in FreemarkerManager	tm_jee	tm_jee	⇧	Resolved	FIXED	Mar 11, 2006	Mar 11, 2006
＋	WW-1233	Add a Plain Text Result	tm_jee	tm_jee	⇧	Resolved	FIXED	Mar 11, 2006	Mar 11, 2006
●	WW-1231	ActionMapper no support for multipart request	Rainer Hermanns	yoshihara hidehiko	⇩	Resolved	FIXED	Mar 10, 2006	Mar 17, 2006
↗	WW-1230	Improve portal integration unit tests	Nils-Helge Garli	Nils-Helge Garli	⇧	Closed	FIXED	Mar 10, 2006	Mar 22, 2006
●	WW-1228	When deploying portlets that use the new webwork	Nils-Helge Garli	Jason Thorpe	⇧	Closed	FIXED	Mar 08, 2006	Mar 17, 2006

		Jsr168Dispatcher class the portlet titles are unavailable in liferay 3.6.1							
↗	WW-1227	Showcase - token examples	Rainer Hermanns	Claus Ibsen	⇩	✓ Closed	FIXED	Mar 05, 2006	Mar 05, 2006
●	WW-1226	Client side JavaScript validation not working when using xwork-tiger	Rainer Hermanns	Rainer Hermanns	⇧	✓ Closed	FIXED	Mar 04, 2006	Mar 18, 2006
↗	WW-1225	Clover report without log debug	Rainer Hermanns	Claus Ibsen	⇩	✓ Closed	FIXED	Mar 04, 2006	Mar 04, 2006
●	WW-1224	Portlet URL support doesn&apos;t follow JSR-168 spec	Nils-Helge Garli	Eric Dalquist	⇧	✓ Closed	FIXED	Mar 03, 2006	Mar 17, 2006
●	WW-1223	div not working without delay > 0 in ajax theme	Ian Roughley	Philip Luppens	⇧	✓ Closed	FIXED	Mar 02, 2006	Mar 16, 2006
↗	WW-1221	Extending xhtml template	Rainer Hermanns	Schava Eugene	⇧	✓ Closed	FIXED	Mar 01, 2006	Mar 17, 2006
●	WW-1220	Some missing javadoc OpenSymphony copyright for recent submitted files	Rene Gielen	Claus Ibsen	⇩	✓ Closed	FIXED	Mar 01, 2006	Mar 02, 2006
+	WW-1219	Create a Button Tag	Rene Gielen	tm_jee	⇧	➜ Resolved	FIXED	Mar 01, 2006	Mar 12, 2006
+	WW-1218	Support <input type=image ...> style html rendering for <ww:submit /> component	Rene Gielen	tm_jee	⇧	➜ Resolved	FIXED	Mar 01, 2006	Mar 17, 2006

●	WW-1217	ww:text tag returns an invalid value for an i18n key that starts with the same name as an action variable	Rainer Hermanns	Nick Hill	⬆	Closed	FIXED	Feb 28, 2006	Mar 17, 2006
●	WW-1216	Action specific I18n texts are not resolved for showcase CRUD example	Rene Gielen	Rene Gielen	⬇	Resolved	CANNOT REPRODUCE	Feb 27, 2006	Mar 14, 2006
↗	WW-1215	ValidationInterceptor - isDebugEnabled() missing	Rainer Hermanns	Claus Ibsen	▼	Closed	FIXED	Feb 26, 2006	Feb 26, 2006
↗	WW-1214	ExecuteAndWaitInterceptor - new features, unit tests, improved javadoc and added examples to showcase	Rainer Hermanns	Claus Ibsen	⬇	Closed	FIXED	Feb 26, 2006	Feb 27, 2006
●	WW-1213	ExecAndWait interceptor - logging spelling error	Alexandru Popescu	Claus Ibsen	▼	Closed	FIXED	Feb 26, 2006	Feb 26, 2006
↗	WW-1212	Avoid excessive usage of ui attributes	tm_jee	tm_jee	⬆	Resolved	FIXED	Feb 25, 2006	Feb 27, 2006
↗	WW-1211	Add sample usage of ww:text tag to showcase example	Rene Gielen	Rene Gielen	⬇	Closed	IMPLEMENTED	Feb 25, 2006	Feb 25, 2006
●	WW-1210	@ww.radio render the same "label for" in multiple form	Patrick Lightbody	Aby Herwendo	⬇	Resolved	FIXED	Feb 24, 2006	Mar 06, 2006

●	WW-1209	The validation is not sent to the client as text/javascript	Patrick Lightbody	Matt Raible	⇧	Resolved	FIXED	Feb 24, 2006	Mar 06, 2006
●	WW-1208	FieldError doesn&apos;t check if fieldError is null before rendering causing freemarker error	tm_jee	tm_jee	⇧	Resolved	FIXED	Feb 24, 2006	Feb 25, 2006
↗	WW-1207	Add anchor attribute to the ww:url tag	Rainer Hermanns	Philip Luppens	⇩	✓Closed	FIXED	Feb 24, 2006	Feb 27, 2006
●	WW-1206	Request encoding mismatch	Rene Gielen	Tobias Jä rlund	⇧	✓Closed	FIXED	Feb 23, 2006	Feb 25, 2006
●	WW-1205	RegexFieldValidator cannot process white space string correctly	Rene Gielen	Robbin Fan	⇧	Resolved	FIXED	Feb 22, 2006	Feb 26, 2006
●	WW-1204	Remove portlet dependancy for Form and URL components	Rainer Hermanns	Rainer Hermanns	⇧	✓Closed	FIXED	Feb 22, 2006	Feb 27, 2006
↗	WW-1202	Improve portal integration javadocs	Nils-Helge Garli	Nils-Helge Garli	⇩	✓Closed	FIXED	Feb 22, 2006	Mar 22, 2006
●	WW-1201	Remote form resultDiv not executing javascript	Ian Roughley	Steve Werner	⇧	✓Closed	FIXED	Feb 21, 2006	Mar 17, 2006
●	WW-1200	richtexteditor is escaping content in ftl	Unassigned	Mike Porter	⇧	Resolved	FIXED	Feb 21, 2006	Feb 21, 2006
↗	WW-1199	Submit tag should support method and action attributes	Patrick Lightbody	Patrick Lightbody	⇧	Resolved	FIXED	Feb 21, 2006	Feb 21, 2006
↗	WW-1198	Use LinkedHashMap	Patrick Lightbody	Patrick Lightbody	⇧		FIXED	Feb 21, 2006	Feb 21, 2006

		instead of TreeMap				Resolved			
●	WW-1196	DoubleSelect&apos;s second select does not allow a predefined value to be selected (WW-1071)	Patrick Lightbody	Alberto Ocampo	⬆	Resolved	FIXED	Feb 21, 2006	Mar 06, 2006
●	WW-1195	labelposition for checkbox-tag does nothing	Don Brown	Peter Westlin	⬇	Resolved	FIXED	Feb 20, 2006	Mar 24, 2006
●	WW-1194	Checkbox-tag without a label generates a Freemarker template error	Rene Gielen	Peter Westlin	⬇	Closed	FIXED	Feb 20, 2006	Feb 26, 2006
●	WW-1193	bind.js calls onLoad asynchronously	Mike Porter	Mike Porter	⬇	Resolved	FIXED	Feb 19, 2006	Feb 19, 2006
●	WW-1192	Generic Support in xwork-tiger.jar does not work	Rainer Hermanns	Corby Page	⬆	Closed	FIXED	Feb 19, 2006	Mar 22, 2006
●	WW-1191	JavaScript validation (non-Ajax) doesn&apos;t look for an existing message before adding a new validation error	Rainer Hermanns	Matt Raible	⬆	Reopened	UNRESOLVED	Feb 18, 2006	Mar 29, 2006
●	WW-1188	select-tag with multiple="false" generates the same HTML as multiple="true"	tm_jee	Peter Westlin	⬇	Resolved	FIXED	Feb 17, 2006	Feb 23, 2006
↗	WW-1187	Allow easy overriding of "excludeMethods" and "includeMethods" of	Rainer Hermanns	Matt Raible	⬆	Resolved	FIXED	Feb 17, 2006	Mar 17, 2006

		ValidationInterceptor							
●	WW-1186	WW 2.2 logs excetion if result of an action is null	Rainer Hermanns	Alex Shneyderman	⬆	Resolved	FIXED	Feb 17, 2006	Feb 20, 2006
↗	WW-1185	Add support for using Tiles with WebWork	Rainer Hermanns	Matt Raible	⬆	Closed	FIXED	Feb 16, 2006	Feb 18, 2006
↗	WW-1183	exclude methods for token interceptor	Alexandru Popescu	Schava Eugene	⬆	Resolved	FIXED	Feb 15, 2006	Mar 17, 2006
●	WW-1182	@ww.token does not work with freemarker result	tm_jee	Claus Ibsen	⬆	Resolved	FIXED	Feb 15, 2006	Feb 26, 2006
↗	WW-1181	Sitemesh freemarker handled (FreemarkerPageFilter.java) should be locale aware	Alexandru Popescu	Christian Meunier	⬇	Resolved	FIXED	Feb 15, 2006	Feb 26, 2006
●	WW-1177	add a debug attribute to ww:head tag	Rainer Hermanns	Mike Porter	⬆	Closed	FIXED	Feb 15, 2006	Feb 27, 2006
●	WW-1176	Ajax elemnts: remote form, remote div,... don&apos;t work on weblogic 8.1	Alexandru Popescu	Adam Cuper	⬆	Resolved	FIXED	Feb 15, 2006	Feb 21, 2006
●	WW-1175	HTTP Response Code wrong for static convent served up via FilterDispatcher	Ian Roughley	Mike Porter	⬆	Resolved	FIXED	Feb 15, 2006	Mar 16, 2006
●	WW-1174	SessionMap does not clear the attributes in the HttpSession	Rainer Hermanns	Dieter van Baarle	⬆	Closed	FIXED	Feb 13, 2006	Feb 20, 2006
●	WW-1173	Freemarker characterset/locale configuration	tm_jee	Dennis Su	⬆	Closed	FIXED	Feb 13, 2006	Feb 18, 2006

		inconsistent with Webwork							
●	WW-1172	submit didn&apos;t include js event	tm_jee	tm_jee	⇧	Resolved	FIXED	Feb 12, 2006	Feb 12, 2006
●	WW-1170	Submit button is always rendered in new line	Alexandru Popescu	Adam Cuper	⇧	Resolved	NOT A PROBLEM	Feb 10, 2006	Feb 17, 2006
⊠	WW-1169	migrate to FilterDispatcher	Rainer Hermanns	victorsosa	⇩	Closed	CANNOT REPRODUCE	Feb 10, 2006	Mar 15, 2006
●	WW-1168	Template / client side validation related files (such as validate.js and style.css) references not resolvable in resulting html	Rene Gielen	Stanley Xu	⇧	Resolved	CANNOT REPRODUCE	Feb 09, 2006	Mar 17, 2006
●	WW-1167	form&apos;s client validation should be executed after user custom submit method	Rene Gielen	Robbin Fan	⇩	Closed	NOT A PROBLEM	Feb 09, 2006	Feb 20, 2006
●	WW-1165	Missing setter for TLD declared attribute &apos;openTemplate&apos; in TreeTag	Rene Gielen	Bhakta Koditipalli	⇧	Resolved	FIXED	Feb 08, 2006	Feb 08, 2006
↗	WW-1164	Remove taglib definition in web.xml for the sample app	Nils-Helge Garli	Nils-Helge Garli	⇩	Closed	FIXED	Feb 08, 2006	Feb 12, 2006
●	WW-1163	Ckeckbox tag fails with freemarker error if no label	Rene Gielen	Rene Gielen	⇧	Resolved	FIXED	Feb 08, 2006	Feb 08, 2006

		set (for xhtml based themes)							
↗	WW-1162	Add support for detecting freemarker errors in UI tags tests	Rene Gielen	Rene Gielen	⬆	Resolved	FIXED	Feb 08, 2006	Feb 08, 2006
●	WW-1159	VelocityPageFilter needs to know about freemarker configuration	Don Brown	Rainer Hermanns	⬆	Resolved	FIXED	Feb 08, 2006	Mar 23, 2006
➕	WW-1158	JasperReport View should support parameters for Report	Rainer Hermanns	Mariusz Smykula	⬆	Closed	NOT A PROBLEM	Feb 07, 2006	Feb 09, 2006
↗	WW-1157	Re-name "action" parameter to avoid possible collission with other parameters	Nils-Helge Garli	Nils-Helge Garli	⬇	Closed	IMPLEMENTED	Feb 07, 2006	Feb 08, 2006
↗	WW-1156	Velocity support	Nils-Helge Garli	Nils-Helge Garli	⬆	Closed	IMPLEMENTED	Feb 07, 2006	Feb 08, 2006
●	WW-1155	DispatcherFilter only reads configuration values for encoding and locale during init - should read from configuration on every request	Rainer Hermanns	Bruce Ritchie	⬆	Closed	FIXED	Feb 07, 2006	Feb 18, 2006
●	WW-1154	Tag ww:text generate Exception related to ww:tree	Rainer Hermanns	Mariusz Smykula	⬆	Closed	DUPLICATE	Feb 07, 2006	Feb 07, 2006
●	WW-1153	Combinations of theme="simple" form components	tm_jee	Ben Rometsch	⬇	Resolved	FIXED	Feb 07, 2006	Feb 12, 2006

		is breaking in certain instances							
●	WW-1152	Theme information lost from parent form tag	tm_jee	Konrad Kamiński	⇧	Resolved	FIXED	Feb 07, 2006	Mar 04, 2006
●	WW-1151	Remote Form submit displays result in new window (IE problem)	Ian Roughley	Adam Cuper	⇑	Resolved	FIXED	Feb 06, 2006	Feb 07, 2006
●	WW-1150	When setting a DatePicker form field as readonly="true" the user can still manipulate the field value	Rene Gielen	Ben Rometsch	⇩	Resolved	FIXED	Feb 06, 2006	Feb 19, 2006
●	WW-1149	Tree tag is has a wrong name in taglib.tld	Rainer Hermanns	Gilles Durys	⇧	Closed	FIXED	Feb 06, 2006	Feb 07, 2006
●	WW-1148	QuickStart does not work on OS X; throws exception	Patrick Lightbody	Stefan Arentz	⇧	Resolved	FIXED	Feb 06, 2006	Mar 06, 2006
↗	WW-1147	Add tooltip support to components	tm_jee	tm_jee	⇧	Resolved	FIXED	Feb 06, 2006	Feb 21, 2006
+	WW-1146	Add a simple RichTextEditor component to webwork	tm_jee	tm_jee	⇧	Resolved	FIXED	Feb 06, 2006	Feb 21, 2006
●	WW-1145	JSTL support not SiteMesh compatible	Rainer Hermanns	Mathias Bogaert	⇧	Resolved	FIXED	Feb 04, 2006	Mar 17, 2006
↗	WW-1144	redirect-action result cannot be configured with GET parameters in	Rene Gielen	Eric Molitor	⇩	Closed	FIXED	Feb 03, 2006	Mar 14, 2006

		xwork.xml							
●	WW-1135	When using the action tag and executeResults in a freemarker template on a jsp page the JSPWriter buffer isn&apos;t being flushed	Rainer Hermanns	Eric Molitor	⇧	✓Closed	FIXED	Feb 01, 2006	Mar 22, 2006
●	WW-1131	Request Parameters are Not Being Decoded	Andres March	Nicholaus Shupe	⇧	✓Closed	NOT A PROBLEM	Jan 29, 2006	Feb 18, 2006
●	WW-1011	Problem with SiteMesh tags and ww:action	Patrick Lightbody	Patrick Lightbody	⇧	✓Closed	FIXED	Dec 14, 2005	Mar 18, 2006
▣	WW-1003	ActionCOnfiguration and xwork.xml overlap	tm_jee	Patrick Lightbody	⇧	➔ Resolved	FIXED	Dec 12, 2005	Mar 21, 2006
↗	WW-959	Access to meta properties in freemaker decorators (without sitemesh tags).	Rainer Hermanns	Kristoffer Skjutare	⇧	✓Closed	FIXED	Nov 17, 2005	Mar 17, 2006
▣	WW-924	Internationalization documents	Rene Gielen	Patrick Lightbody	⇧	✓Closed	FIXED	Oct 27, 2005	Mar 22, 2006
▣	WW-852	Document complete Spring integration	Rainer Hermanns	Patrick Lightbody	⇧	➔ Resolved	FIXED	Oct 07, 2005	Mar 16, 2006
●	WW-822	Setting the character encoding on the request causes all form parameters to be erased on Oracle 9i	Rene Gielen	accbank	⇩	✓Closed	CANNOT REPRODUCE	Sep 03, 2005	Mar 14, 2006
●	WW-808	ParamTag does not convert to string	Claus Ibsen	John Patterson	⇩	✓Closed	FIXED	Jul 15, 2005	Mar 18, 2006
➕	WW-805	ww:date tag	Rainer Hermanns	Patrick Lightbody	⇧	✓Closed	FIXED	Jul 12, 2005	Mar 22, 2006

	WW-724	FileUpload Error [File Type Tiff , PDF] or [network Drive File]	Rainer Hermanns	toshihito nogami	⇧	Resolved	CANNOT REPRODUCE	Jan 18, 2005	Feb 18, 2006
	WW-649	Field errors should be displayed in the order they&apos;re in the POJO	tm_jee	Matt Raible	⇧	Resolved	FIXED	Sep 29, 2004	Mar 03, 2006

3rd Party Integration

This page last changed on Mar 30, 2006 by scud.

1. Sitemesh
2. Spring
3. Pico
4. Hibernate
5. JSTL
6. JUnit
7. Quartz
8. JasperReports

Hibernate

This page last changed on Mar 30, 2006 by scud.

There's nothing more that you have to do use Hibernate with WebWork than with other Web framework. Just setup Hibernate according to the http://www.hibernate.org/5.html. However, there're a number of good patterns that people have used successfully in the following projects:

- AdminApp http://www.hibernate.org/159.html#a5
- Petsoar http://www.wiley.com/legacy/compbooks/walnes

- Non\-IoC version of OpenSessionInViewInterceptor by Gary

AdminApp

This page last changed on May 19, 2006 by scud.

TODO: Documenting updated version (currently at http://www.i-tao.com/adminapp.html). IN PROGRESS.

Introduction

This page aims at providing some additional information about the Hibernate AdminApp. The Hibernate AdminApp (hereafter referred to as AA) was created by the Hibernate developers to show a possible implementation strategy for Hibernate with Webwork. Although AA can still be used as a starting point for webapplications, most of its libraries become quite aged (WW 2.0 beta, Hibernate 2, XWork 1.0 beta). Therefore, a shiny new fork (AA2) has been created by Ron Chan.

AA2 relies on WW2.2, Hibernate 3.1, and Spring as its IoC container (rather than XWork's, which has been deprecated in WW 2.2). We'll first discuss the original AA. Later on, we'll show the differences with AA2. Ron, if you're reading this, feel free to point out any mistakes/edit this document.

Like we pointed out before, AA shows a possible implementation strategy to use Hibernate in WebWork in combination with a so-called open-session-in-view pattern (more info, even more). This pattern allows maximum flexibility in our view layer by creating a Hibernate Session object that will live till the end of the request (after the view has been rendered). This allows lazy loading of objects in our view, rather than having to preload all objects and their associations in our business layer, and yet ensures the correct disposing of the Session object.

To accomplish this, AA uses XWork's components and interceptors:

- components: XWork manages the lifecycle of objects in several scopes (application, session, request) and takes care of the IoC through the ..Aware interfaces (so called enablers). Hibernate's expensive-to-create SessionFactory will thus be created in the application scope (meaning it will only be initialised once when the application starts up), while the Session objects, used to load our models, is registered in the request scope (will be created once per request).

- interceptors: AA uses an interceptor (the HibernateInterceptor) to extract the Session from the WebWork action, so it can control the transactions, redirect/rollback on errors and properly dispose the Session after the view is rendered.

AdminApp Source Overview

Now, let's properly dissect the AA files:

- /lib: contains the various jars for our application. Nothing special here.
- /src/java/org/hibernate/admin/action: lists our WebWork actions. All actions extend an abstract AbstractAction file, which overrides the execute() method from our XWork's ActionSupport. This is where we define a setHibernateSession() method, which is the method we declared in our enabler interface (HibernateSessionAware). This will notify XWork to invoke its IoC magic to set the HibernateSession.

```
public String execute() throws Exception {

                // We go to INPUT on field and data errors
        if ( hasErrors() ) {
                        LOG.debug("action not executed, field or action errors");
                        LOG.debug( "Field errors: " + getFieldErrors() );
                        LOG.debug( "Action errors: " + getActionErrors() );
                        return INPUT;
                }

                LOG.debug("executing action");
                return go();
        }

        protected abstract String go() throws HibernateException;

        public void setHibernateSession(HibernateSession session) {
                this.session = session;
        }

        protected Session getSession() throws HibernateException {
                return session.getSession();
        }
```

In this execute() method we'll simply call a abstract go() method (which is then defined in each of the actions). When we need the Hibernate Session, we use the getSession() method, inherited from our AbstractAction. Don't worry about transactions or saving so called dirty objects (our HibernateInterceptor takes care of all that). As you can see, this totally minimizes the LoC (lines of code) needed to retrieve or manipulated our models).

```
public class EditUserAction extends AbstractAction {
        //.. ommited for brevity

        protected String go() throws HibernateException {
                ..
                getSession().update(user);
                ..
                return SUCCESS;
        }

        //.. getters and setters ommited

}
```

There are 3 more *-validation.xml files in this directory containing the validation logic for the Actions. XWork will validate your request before the action gets executed, so you can decouple your (simple) validation logic from your Action. For example, take a look at the CreateUserAction-validation.xml:

```
..
    <field name="user.name.lastName">
        <field-validator type="requiredstring">
            <message>You must enter a last name.</message>
        </field-validator>
    </field>
    <field name="user.email">
        <field-validator type="email">
            <message>Please correct the e-mail address.</message>
        </field-validator>
        <field-validator type="required">
            <message>Please enter an e-mail address.</message>
        </field-validator>
    </field>
..
```

Several validator types are available. Here we rely on XWork to validate our Actions, but it's also possible to validate our object Models (see WW Validation). You will mostly use these to validate submitted forms in your webapp.

When a validator fails, you will automatically be returned to the input page with a clear indication which field failed to validate if:

a) actually provided an input type in your xwork.xml file

```
..
        <result name="input" type="dispatcher">
                <param name="location">/editUser.jsp</param>
        </result>
        ..
```

b) you enabled the validation interceptor in your xwork.xml

```
..
        <interceptor-ref name="defaultStack"/>
        <interceptor-ref name="validation"/>
        ..
```

c) you use the WebWork tag library (warning: this is the old syntax):

```
..
<ww:form name="'createUserForm'" action="'createUser.action'" method="'POST'">
    <ww:textfield label="'Username'" name="'user.handle'"/>
..
```

New syntax (since 2.2):

```
..
<ww:form name="createUserForm" action="createUser" method="POST">
    <ww:textfield label="Username" name="user.handle"/>
..
```

- /src/java/org/hibernate/admin/component: contains the components and enablers for both the HibernateSessionFactory and the HibernateSession. These components are declared in the /src/java/components.xml file (which will be copied to the root of your compiled classes afterwards):

```
<components>

    <component>
        <scope>request</scope>
        <class>org.hibernate.admin.component.HibernateSession</class>
        <enabler>org.hibernate.admin.component.HibernateSessionAware</enabler>
    </component>

    <component>
        <scope>application</scope>
        <class>org.hibernate.admin.component.HibernateSessionFactory</class>
        <enabler>org.hibernate.admin.component.HibernateSessionFactoryAware</enabler>
    </component>

</components>
```

- /src/java/org/hibernate/admin/interceptor: contains the Hibernate interceptor. Interceptors are an incredibly powerful feature of WebWork - it allows you to control invocations before and after they excute, manipulate their results, or, as in our case, extract the HibernateSession object and dispose it after the Action has been executed (and the view rendered). Because we use a try/catch/finally block, we're able to catch exceptions and yet make sure our Session gets closed properly (the number one cause of db connection leaks).

```
  public String intercept(ActionInvocation invocation) throws Exception {
                Action action = invocation.getAction();
                if ( !(action instanceof AbstractAction) ) return invocation.invoke();

                HibernateSession hs = ( (AbstractAction) action ).getHibernateSession();
                try {
                        return invocation.invoke();
                }

                // Note that all the cleanup is done
         // after the view is rendered, so we
         // have an open session in the view

                catch (Exception e) {
                        hs.setRollBackOnly(true);
                        if (e instanceof HibernateException) {
                                LOG.error("HibernateException in execute()", e);
                                return Action.ERROR;
                        }
                        else {
                                LOG.error("Exception in execute()", e);
                                throw e;
                        }
                }

                finally {
                        try {
                                hs.disposeSession();
                        }
                        catch (HibernateException e) {
                                LOG.error("HibernateException in dispose()", e);
                                return Action.ERROR;
                        }
                }
        }
```

Conclusion

In this document, we tried to point out several key features in the Hibernate AdminApp. In part II, we'll have a look at the new AdminApp, which is far more up to date, and uses Spring as its IoC container. No more implements ActionSupport or Aware interfaces, resulting in even cleaner code.

AdminApp is a very good example of how a webapp can be structered, using as many advantages from the various frameworks as possible.

Non-IoC version of OpenSessionInViewInterceptor

This page last changed on Mar 30, 2006 by scud.

Gary was so kind to provide us a non-IoC Hibernate 'Open Session in View'-interceptor. Rather than having XWork or Spring doing the dependency injection, he sets up the Hibernate Session himself.

```
/*
  * HibernateOpenSessionInViewInterceptor.java
  *
  * Created on March 18, 2006, 3:51 PM
  *
  * To change this template, choose Tools | Template Manager
  * and open the template in the editor.
  */

package edu.washington.javawebdevelopment.webwork.interceptor;

import com.opensymphony.xwork.ActionInvocation;
import com.opensymphony.xwork.interceptor.AroundInterceptor;
import edu.washington.javawebdevelopment.dao.DaoFactoryHibernate;
import javax.servlet.ServletException;
import org.hibernate.SessionFactory;
import org.hibernate.StaleObjectStateException;

/**
  *
  * @author gary
  */
public class HibernateOpenSessionInViewInterceptor extends AroundInterceptor {
     private SessionFactory hibernateSessionFactory;

     public void init() {
         System.out.println("Initializing HibernateOpenSessionInViewInterceptor interceptor,
obtaining Hibernate SessionFactory from DaoFactoryHibernate");
         hibernateSessionFactory = DaoFactoryHibernate.getSessionFactory();
     }

     public void destroy() {
     }

     public void before(ActionInvocation invocation) throws Exception {
         System.out.println("Starting a database transaction in the
HibernateOpenSessionInViewInterceptor");
         hibernateSessionFactory.getCurrentSession().beginTransaction();
     }

     public void after(ActionInvocation invocation, String result) throws Exception {
         // Commit and cleanup
         try {
             System.out.println("Committing the database transaction in the
HibernateOpenSessionInViewInterceptor");
             hibernateSessionFactory.getCurrentSession().getTransaction().commit();
         } catch (StaleObjectStateException staleEx) {
             System.err.println("This interceptor does not implement optimistic concurrency
control!");
             System.err.println("Your application will not work until you add compensation
actions!");
             // Rollback, close everything, possibly compensate for any permanent changes
             // during the conversation, and finally restart business conversation. Maybe
             // give the user of the application a chance to merge some if his work with
             // fresh data... what you do here depends on your applications design.
             throw staleEx;
         } catch (Throwable ex) {
             // Rollback only
             ex.printStackTrace();
             try {
                 if (hibernateSessionFactory.getCurrentSession().getTransaction().isActive()) {
                     System.out.println("Trying to rollback database transaction after
exception");
                     hibernateSessionFactory.getCurrentSession().getTransaction().rollback();
                 }
             } catch (Throwable rbEx) {
                 System.err.println("Could not rollback transaction after exception! - " +
rbEx);
             }

             // Let others handle it... maybe another interceptor for exceptions?
             throw new ServletException(ex);
         }
     }
}
```

JSTL

This page last changed on Mar 30, 2006 by scud.

JSTL integration is built in to WebWork 2.2+ - there are no steps required to enable it. Simply refer to your JSTL expressions just as you would with a normal WebWork JSP tag, such as the property tag. This is accomplished a request wrapper called For more info, see the javadocs of the WebWorkRequestWrapper object:

All WebWork requests are wrapped with this class, which provides simple JSTL accessibility. This is because JSTL works with request attributes, so this class delegates to the value stack except for a few cases where required to prevent infinite loops. Namely, we don't let any attribute name with "#" in it delegate out to the value stack, as it could potentially cause an infinite loop. For example, an infinite loop would take place if you called: request.getAttribute("#attr.foo").

Pico

This page last changed on Mar 30, 2006 by scud.

Pico is an Inversion of Control container available at http://picocontainer.codehaus.org. There have been several reports of integration between WebWork and Pico. As of WebWork 2.2, Pico integration is built in. To use Pico, you must first install the Pico Dependencies. Then, instead of using the normal filter dispatcher in web.xml that maps to /*, you simply need to use the class `com.opensymphony.webwork.pico.PicoFilterDispatcher`.

For WebWork versions previous to 2.2, http://www.nanocontainer.org/NanoWar+WebWork contains information on integrating WebWork and Pico/Nano.

Quartz

This page last changed on Mar 30, 2006 by scud.

The following class performs the glue between Quartz and WebWork:

```
package com.trantek.sit.action;

import com.opensymphony.xwork.ActionProxy;
import com.opensymphony.xwork.ActionProxyFactory;
import com.opensymphony.xwork.interceptor.component.ComponentInterceptor;
import org.quartz.Job;
import org.quartz.JobExecutionContext;
import org.quartz.JobExecutionException;
import java.util.HashMap;

public class WebWorkJob implements Job
{
    public void execute(JobExecutionContext context) throws JobExecutionException
    {
        try
        {
            HashMap ctx = new HashMap();
            ctx.put(ActionContext.PARAMETERS, context.getJobDetail().getJobDataMap());
            ctx.put(ComponentInterceptor.COMPONENT_MANAGER, ???);
            ctx.put(???, ???)
            ServletDispatcher.createContextMap()
            ActionProxy proxy = ActionProxyFactory.getFactory().
                    createActionProxy("", context.getJobDetail().getName(), ctx);

            proxy.execute();
        }
        catch (Exception e)
        {
            throw new JobExecutionException(e);
        }
    }
}
```

To schedule webwork actions you simply create a job where

- the name of your job is the name of the WW action to execute (no ".action" suffix).
- all the parameters you want to send to the WW action is contained in the JobDataMap of the JobDetail

(the Quartz scheduler is setup as a servlet according to the javadocs of org.quartz.ee.servlet.QuartzInitializerServlet.)

The following code schedules an e-mail action:

```
Scheduler scheduler = StdSchedulerFactory.getDefaultScheduler();

JobDetail jobDetail = new JobDetail("email.send",
                                    scheduler.DEFAULT_GROUP, WebWorkJob.class);

Map m = jobDetail.getJobDataMap();
m.put("to", "me@bogusdomain.com");
m.put("subject", "quartz test");
m.put("body", "This is a quartz test, Hey ho");
m.put("smtpServer", "smtp.bogusdomain.com");
m.put("from", "quartz@bogusdomain.com");

SimpleTrigger trigger = new SimpleTrigger("myTrigger",
                                          scheduler.DEFAULT_GROUP,
                                          new Date(), null, 0, 0L);

scheduler.deleteJob("email.send", scheduler.DEFAULT_GROUP);
```

```
    scheduler.scheduleJob(jobDetail, trigger);
```

This example is based on [WW1:Integrating Webwork and Quartz]

SiteMesh

This page last changed on May 22, 2006 by foxcoming.

Overview

SiteMesh是一个Web页面布局修饰框架，用于构建包含大量页面，需要一致的外观样式(look/fell)，导航和布局机制的大型网站.

在WebWork中集成SiteMesh相当容易：实际上什么也不用做. WebWork把全部值栈数据都保存在请求attribute中，这意味着如果想显示值栈(或ActionContext)中的数据，只需使用WebWork附带的标准标签库，就这么简单！

ActionContextCleanUp

在WebWork的架构中，标准的过滤器链(filter-chain)一般以 `ActionContextCleanUp` 开始，后面跟着其他需要的过滤器. 最后，由 `FilterDispatcher` 处理请求，通常是将请求传递给ActionMapper. `ActionContextCleanUp` 的首要用途是为集成SiteMesh服务的. 他会通知FilterDispatcher在正确的时间清除请求. 否则，ActionContext将在SiteMesh修饰器访问数据之前被清除.

警告
如果需要在修饰器中访问ActionContext，`ActionContextCleanUp` 过滤器必须放在过滤器链的起点.

更多信息参见ActionContextCleanUp的JavaDoc文档：

该过滤器用来与FilterDispatcher协同工作以便于集成SiteMesh. 通常，排列过滤器并保证SiteMesh排在第一位，而让FilterDispatcher排在第二位看起来是最佳方案. 然而，你可能希望在SiteMesh修饰器中使用WebWork的特性，包括value stack，但由于FilterDispatcher将清除ActionContext，因此修饰器就访问不到想要的数据了.

通过添加本过滤器，FilterDispatcher会知道先不执行清除而是将清除推迟到本过滤器中执行，修改后的顺序如下：

- 本过滤器
- SiteMesh
- FilterDispatcher

(摘自snippet:id=description|javadoc=true|url=com.opensymphony.webwork.dispatcher.ActionContextCleanUp)

Velocity and FreeMarker Decorators

WebWork提供了SiteMesh的PageFilter的一个扩展版本的过滤器用于协助与Velocity和FreeMarker的集成. 强烈推荐使用这个过滤器来取代SiteMesh提供的过滤器，因为当你使用自己喜欢的模版语言创建视图时，也能够使用标准的变量和标签.

Velocity

如果你要使用Velocity作为SiteMesh修饰器，推荐使用VelocityPageFilter. 这是SiteMesh的PageFilter的一个扩展版本，应该定义在web.xml中，位于 `ActionContextCleanUp` 和 `FilterDispatcher` 之间. 这样Velocity修饰器就可以使用$stack和$request来访问WebWork的变量.

```
<filter>
```

```
        <filter-name>webwork-cleanup</filter-name>
        <filter-class>com.opensymphony.webwork.dispatcher.ActionContextCleanUp</filter-class>
    </filter>
    <filter>
        <filter-name>sitemesh</filter-name>
        <filter-class>com.opensymphony.webwork.sitemesh.VelocityPageFilter</filter-class>
    </filter>
    <filter>
        <filter-name>webwork</filter-name>
        <filter-class>com.opensymphony.webwork.dispatcher.FilterDispatcher</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>webwork-cleanup</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>sitemesh</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>webwork</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

FreeMarker

如果你要使用FreeMarker作为SiteMesh修饰器, 推荐使用FreeMarkerPageFilter. 这是SiteMesh的PageFilter的一个扩展版本, 应该定义在web.xml中, 位于 ActionContextCleanUp 和 FilterDispatcher 之间. 这样FreeMarker修饰器就可以使用${stack}和${request}.来访问WebWork的变量.

```
    <filter>
        <filter-name>webwork-cleanup</filter-name>
        <filter-class>com.opensymphony.webwork.dispatcher.ActionContextCleanUp</filter-class>
    </filter>
    <filter>
        <filter-name>sitemesh</filter-name>
        <filter-class>com.opensymphony.webwork.sitemesh.FreeMarkerPageFilter</filter-class>
    </filter>
    <filter>
        <filter-name>webwork</filter-name>
        <filter-class>com.opensymphony.webwork.dispatcher.FilterDispatcher</filter-class>
    </filter>

    <filter-mapping>
        <filter-name>webwork-cleanup</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>sitemesh</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
    <filter-mapping>
        <filter-name>webwork</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

Spring

This page last changed on May 19, 2006 by scud.

伴随着其他东西,Spring同时也是一个控制反转框架.(原文:Spring is, among other things, an Inversion of Control framework.) 它在WebWork 2.2中是推荐的IoC容器.你可以在http://www.springframework.org找到更多关于Spring的信息.

此部分仅包括 支持 Spring集成的技术.当然,还有很多其他的方法把Spring和WebWork绑定在一起.请查看其他集成Spring的方法了解更多信息. 注意其他方法的任何一个都不被支持,而且可能随时发生变化!(原文:Note that none of the other methods are currently supported and could change at any time!)

开启Spring集成

在WebWork中开启Spring支持是一件简单的事情,只需要安装最新的Spring的jar文件到你的classpath,然后添加下面的行到webwork.properties文件:

```
webwork.objectFactory = spring
```

如果你想要改变缺省的自动装配模式,也就是自动按照name装配(例如,使用你的bean属性相同的名字去查找在Spring里定义的bean),你也需要在你的webwork.properties中进行一下设置:

```
webwork.objectFactory.spring.autoWire = type
```

这个设置的选项包括:

name	按照你的action的属性的名字和Spring里的bean的名字,如果匹配就自动装配. **这是缺省的**
type	按照你的action的属性的类型,在Spring注册的bean中查找,如果相同就自动装配.这需要你在Spring中仅注册了一个此类型的bean
auto	Spring会试图自动监测来找到最好的方法自动装配你的action
constructor	Spring会自动装配bean的构造函数的参数

在这种情况下,所有的对象都至少会试图使用Spring来创建.如果它们不能被Spring创建,然后WebWork会自己创建对象.接下来,你需要在web.xml打开Spring的Listener:

```
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

需要更多的 ApplicationContext 配置文件?

既然使用一个标准的Listener来集成Spring,它可以被配置来支持除了applicationContext.xml之外的配置文件.把下面的几行添加到你的web.xml会让Spring的ApplicationContext从所有匹配给定的规则的文件中初始化:

```
<!-- Context Configuration locations for Spring XML files -->
<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>/WEB-INF/applicationContext-*.xml,classpath*:applicationContext-*.xml</param-value>
</context-param>
```

浏览Spring的文件查看这个参数的详细描述.

Spring配置举例

在这里,你可以添加标准的Spring配置文件 WEB-INF/applicationContext.xml .这个配置文件的一个例子如下:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans default-autowire="autodetect">
    <bean id="personManager" class="com.acme.PersonManager"/>
    ...
</beans>
```

从内置的IoC切换到Spring

切换是非常简单的.Spring的设置就像上面描述的已经完成.为了完成迁移,你还需要
Switching is quite easy. Spring setup is done as described above. To complete migration, you will have to

1. 适当地转换你配置的组件,从 components.xml 到 applicationContext.xml. 然后你可以安全地删除 components.xml.
2. 在xwork.xml中,从你的拦截器stack里移除 组件拦截器. 虽然保留它在那里不会带来任何影响,但是从现在开始它只不过是多余的了.

Session 会话范围和Spring

Spring <= 1.3 不支持 session 会话范围的组件. Spring 2.0 版本会增加这个支持, 并且在对Spring 2.0M3 (beta) 的测试中被报告可以工作的很好. 请查阅 和Spring的Session组件一起工作 来了解这个话题的更详细信息.

从Spring中初始化 Action

正常情况下,在xwork.xml里你可以为每个action指定类.当你使用SpringObjectFactory(就像上面配置的)时WebWork会请求Spring来创建action并按照缺省指定的自动装配行为来装配依赖的组件.SpringObjectFactory 也会设置所有的bean的后置处理程序(post processors)来完成类似对你的Action进行事务,安全等等方面的代理的事情.Spring可以不依赖外在的配置来自动确定.对于大多数的使用,这就是你所全部需要的了,用来配置你的action,设置它们获取服务和依赖组件.

我们 **强烈** 推荐你找到一种声明式的方法来让Spring知道为你的action提供什么.这包括让你的bean能够自动装配,无论是把你的Action里的依赖的属性命名为和Spring应该提供的Bean的名字一致(这允许基于名字的自动装配),或者使用by type方式的自动装配,也就是在注册到Spring的Bean中需要的类型仅拥有一个.也可以包括使用JDK5的标准来声明事务和安全需求,而不是必须在你的Spring配置里明确设置代理.如果你能找到方法让Spring在没有任何明确的配置(在_applicationContext.xml_中)的情况下知道需要为你的action做什么,那么你就不需要在两个地方维护这个配置了.

However, sometimes you might want the bean to be completely managed by Spring. This is useful, for example, if you wish to apply more complex AOP or Spring-enabled technologies, such as Acegi, to your beans. To do this, all you have to do is configure the bean in your Spring applicationContext.xml and then change the class attribute from your WebWork action in the xwork.xml to use the bean name defined in Spring instead of the class name.

当然,有时候你可能想要Spring完全来管理bean.这是有实际意义的,例如,如果你想要为你的bean设置更复杂的AOP或者Spring相关的技术,例如Acegi.为了达到这个目的,你所有必须要做的事情就是在Spring的 applicationContext.xml 里配置你的bean,然后在 xwork.xml 里改变你的WebWork action的类属性来使用在Spring里面定义的bean的名字,而不再使用类名.

你的xwork.xml文件也会改变action类的属性,最后留下的就像这样:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.1.dtd">

<xwork>
    <include file="webwork-default.xml"/>

    <package name="default" extends="webwork-default">
        <action name="foo" class="com.acme.Foo">
            <result>foo.ftl</result>
        </action>
    </package>

    <package name="secure" namespace="/secure" extends="default">
        <action name="bar" class="bar">
            <result>bar.ftl</result>
        </action>
    </package>
</xwork>
```

在你的 applicationContext.xml 里定义了一个名字为 "bar"的Spring的bean.注意 com.acme.Foo 不需要改变,因为它可能是自动装配的.

一个典型的spring的bar的配置可能看起来像下面列出的一样.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans default-autowire="autodetect">
    <bean id="bar" class="com.my.BarClass" singleton="false"/>
    ...
</beans>
```

注意spring配置里面的id属性对应xwork配置里面的class属性. 还要注意在spring配置中,singleton(单态) 属性被设置为false. 这通常是因为WebWork希望基于每个request来创建一个新的action.因此当用到Spring集成时,这会是期望的行为.让Spring的singleton 属性为false就可以做到这一点.

记住: **这不是必须的.** 仅仅当你希望覆盖在WebWork中创建一个action时的缺省行为,而要用Spring的拦截器,以及Spring不能自动确定IoC策略时,这才是必要的. 要记住一点,WebWork的Spring集成会进行标准的IoC,使用你指定的任何自动装配策略,即使你没有明确第在Spring中映射每个Action.所以通常你不需要做这个,但是知道怎么做当你需要的时候对你是有好处的.

Other Spring Integration

This page last changed on Apr 24, 2006 by scud.

译者注:此页不进行翻译了,此页大部分情况已经不适合WebWork2.2了,当然对于WebWork 2.1.7还是有用的. 不过已经有过很多文章写WebWork 2.1.7结合Spring了.

This document is provided by the WebWork user community and does not represent the supported Spring integration methods. Please refer to Spring for documentation on the recommended integration.

I started out using the original WebWork documentation to get Spring to initialize Webwork actions, but it appears that a lot has changed since the days of WebWork 1.x. This will be my attempt to clarify some of those changes and to list the steps necessary to get the two to play nicely. Please comment with clarifications or corrections!

WebWork 1.x

(these are assumptions based on the 1.x [WW1:Spring Framework Integration] documentation)
It seems that the way you got Spring to initialize WebWork 1.x action classes was to add this line into your webwork.properties file:

```
webwork.action.factory=webwork.action.factory.SpringActionFactory
```

so that WebWork would know to use Spring's view of the world to create actions. The WebWork action classes would then need to be declared in the Spring applicationContext.xml file so that Spring would know directly of the action objects. Upon invocation of an action, WebWork would know to first use the SpringActionFactory to try and create an instance of the requested action which would ask Spring to create the object using its configuration. If there was no Spring definition of that action object, then WebWork would use it's normal instantiation methods to create that action. Well, things have changed slightly since WebWork 1.x.

WebWork 2

In WebWork 2 (the functionality actually exists in XWork), you specify relationships from action classes to other objects in XWork's xwork.xml file instead of Spring's applicationContext.xml file. So if you have an action class that utilizes a DAO, instead of having a bean definition like so in applicationContext.xml:

```
<bean id="myAction" class="com.ryandaigle.web.actions.MyAction" singleton="false">
        <property name="DAO">
                <ref bean="myDAO"/>
        </property>
<bean id="myDAO" class="com.ryandaigle.persistence.MyDAO" singleton="true" />
```

you move the action definition to xwork.xml and keep the DAO definition in applicationContext.xml so that xwork.xml looks like:

```
<action name="myAction" class="com.ryandaigle.web.actions.MyAction">
        <external-ref name="DAO">myDAO</external-ref>
        <result name="success" type="dispatcher">
                <param name="location">/success.jsp</param>
        </result>
```

```
</action>
```

and applicationContext.xml looks like:

```
<bean id="myDAO" class="com.ryandaigle.persistence.MyDAO" singleton="true" />
```

Notice how there is the external-ref element in the action definition that points to an object that Spring is managing. There are several things that need to be in place for the external-ref to work, but I just wanted to give an overview of what has changed before going into the specific steps.

Steps for Configuring Spring/WebWork2 (XWork) Integration:

Get the files you need to externally resolve Spring beans. I've bundled them all here: http://www.ryandaigle.com/pebble/images/webwork2-spring.jar . They were originally spread between two JIRA issues filed against XWork 1.0 (see references below). This zip includes the source, the class files (so you can just include it in your classpath) and my example configuration files. Either extract the source files into your application, or put the file onto your classpath. (You may want to take the applicationContext.xml and xwork.xml files out, I don't know if they'll override your files... They're just there as an example configuration).

Now, let's get your XWork configuration file (xwork.xml) to resolve external references. XWork resolves external references (using the external-ref element) by utilizing an external reference resolver per package. You specify your external reference resolver as an attribute of the package element:

```
<package name="default" extends="webwork-default"
        externalReferenceResolver="com.atlassian.xwork.ext.SpringServletContextReferenceResolver">
```

This SpringServletContextReferenceResolver class reference is a class not part of the XWork distribution written as an extensions for XWork/Spring that I got from this JIRA issue filed against XWork addressing this Spring integration effort. (I have bundled it with the rest of the necessary files later on down for your convenience). This class will intercept all external-refs and resolve the references using Spring's context. There is also a SpringApplicationContextReferenceResolver included in the zip file that will allow you to resolve Spring references for applications not executing within the web context. But as this is a WebWork/Spring article, the servlet resolver is what we need to use.

Now we need to add the XWork reference resolver as part of the interceptor stack you're using. This will allow any references to be resolved (using the reference resolver you specified in the externalReferenceResolver attribute). This is how I've added that interceptor:

```
<interceptors>
        <interceptor name="reference-resolver"
class="com.opensymphony.xwork.interceptor.ExternalReferencesInterceptor"/>
        <interceptor-stack name="myDefaultWebStack">
                <interceptor-ref name="defaultStack"/>
                <interceptor-ref name="reference-resolver"/>
        </interceptor-stack>
</interceptors>
<default-interceptor-ref name="myDefaultWebStack"/>
```

As I briefly outlined before, you can now reference Spring beans that your action classes need in xwork.xml:

```
<action name="myAction" class="com.ryandaigle.web.actions.MyAction">
```

```
        <external-ref name="DAO">myDAO</external-ref>
        <result name="success" type="dispatcher">
                <param name="location">/success.jsp</param>
        </result>
</action>
```

And that's all we have to do to xwork.xml to let XWork know how to resolve references to Spring's managed beans.

Now let's setup our web environment to properly notify Spring and our external reference resolver of the web context. We do this by adding two context listeners to your application's web.xml file:

```
<listener>
        <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<listener>
        <listener-class>com.atlassian.xwork.ext.ResolverSetupServletContextListener</listener-class>
</listener>
```

The first listener is Spring's that you would need independent of whether or not you were integrating with WebWork 2. The second listener is our external resolver's that will use the servlet context to retrieve Spring's application context. This is the link between WebWork and Spring.
At this point, we've set up Spring and our XWork reference resolver to work within a web context, and we've told XWork how to resolve external references to Spring. We're done! Fire it up and let me know if there are some steps I've missed or assumptions I've made that I shouldn't have.

References

- Here is my bundled source, class and example configuration files (that contains all the needed referenced files below); http://www.ryandaigle.com/pebble/images/webwork2-spring.jar .
- My searching started with the original WebWork 1.x + Spring documentation and comments on the Wiki; [WW1:Spring Framework Integration]
- The Wiki pointed me to the two JIRA issues that contained the source files for the reference resolvers:
- http://jira.opensymphony.com/browse/XW-122 The "SpringExternalResolver.zip" attachment is the one needed for externally resolving Spring objects.
- http://jira.opensymphony.com/browse/XW-132 The "xwork-springServletImpl.zip" attachment is the one needed for externally resolving Spring objects. It just contains some files missing from the original source.

Credits

Judging by the comments etc... of the JIRA issues filed against XWork, it appears that Ross Mason (of Atlassian?) is the man to thank for the external reference resolver code. And of course we have to thank the people of Spring and WebWork 2 for making this all possible.

Using the SpringObjectFactory

Rather than using an external reference resolver with releases of XWork from 1.0.1 and onwards, it's possible to use the SpringObjectFactory from the xwork-optional" package. This uses Spring to wire up the dependencies for an Action before passing it to XWork. Each action should be configured within a Spring application context as a prototype (because XWork assumes a new instance of a class for every action invocation):

```
<bean name="some-action" class="fully.qualified.class.name" singleton="false">
    <property name="someProperty"><ref bean="someOtherBean"/></property>
</bean>
```

Within xwork.xml:

```
<action name="myAction" class="some-action">
    <result name="success">view.jsp</result>
</action>
```

Notice that the XWork Action's class name is the bean name defined in the Spring application context.

The 1.1.3 release of the Spring/XWork integration library allows the user to configure everything in the `xwork.xml` file without needing to add extra entries to the `applicationContext.xml`. This is done by configuring the actions with the fully qualified class name (as if not using the SpringObjectFactory) It also added the ability to make use of constructor-based dependency injection without any further changes. The major caveat when using constructor-based DI is that objects passed in to the constructor must be unambiguous within the applicationContext (as is normally required by Spring) If there is any ambiguity, then you can still configure things the more traditional way, splitting the configuration of the action between `xwork.xml` and `applicationContext.xml` as described above.

One other advantage of the SpringObjectFactory approach is that it can also be used to load interceptors using the same sort of logic. If the interceptor is stateless, then it's possible to create the interceptor as a singelton instance, but otherwise it's best to create it as a Spring prototype.

In order to be used, the default ObjectFactory that XWork uses should be replaced with an instance of the SpringObjectFactory. The xwork-optional package ships with a ContextListener that does this, assuming that the Spring application context has already been configured.

ActionAutowiringInterceptor

Another alternative to using the SpringObjectFactory is to use the ActionAutowiringInterceptor. The interceptor will autowire any action class based on the autowire strategy defined. An advantage to using the interceptor over the SpringObjectFactory is that the action classes do not have to defined in the Spring's application context. The following is an example of how it can be configured in xwork.xml:

```
<interceptors>
  <interceptor name="autowire"
class="com.opensymphony.xwork.spring.interceptor.ActionAutowiringInterceptor">
    <param name="autowireStrategy">1</param>
  </interceptor>
  <interceptor-stack name="autowireDefault">
    <interceptor-ref name="autowire"/>
    <interceptor-ref name="defaultStack"/>
  </interceptor-stack>
</interceptors>
```

Note the the autowireStrategy parameter is optional. If you do not define it, then the SpringObjectFactory will default to autowiring by name. The interceptor looks for Spring's application context in the XWork's application context. To initialize the application context, add the following listener to your web.xml:

```
<listener>
  <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

You do not have to configure the SpringObjectFactory seperately unless you plain on instantiating results, interceptors, or validators as Spring beans. As a convenience method to get access to the application context for other uses, it is placed in the ActionContext map under the key ActionAutowiringInterceptor.APPLICATION_CONTEXT for each Action.

Spring Session Components Workarounds

This page last changed on Apr 24, 2006 by scud.

动机

Spring现在不支持session会话范围的bean和component.你可以在singleton或者prototype生命周期之间选择,但是你不能让你的bean绑定到web应用程序的session生命周期上去.在Spring 2.0版本中已经计划集成这样一个特性.我们试着指出一些可能的解决方案来工作在基于WebWork的应用程序中.首先我们会考查在Spring社区中发现的通用解决方案,来处理HTTPSession和诸如此类的事情.然后我们会讨论在XWork/WebWork里发现的特殊情况和需求,以及这些会如何影响可能的解决方案.我们会展示一些XWork/WebWork对给定问题的特定解决方案.

Spring 2.0(前一个版本是1.3)的第一个里程碑版本将会在2005十二月的第二个星期发布.已经证实它会包含一个Session会话范围的组件,使用了Proxy(CGLIB或者JDK)方法.

对于web应用程序的通用解决方案

Spring 2.0 的方法

Interface21(译注:Spring作者的公司)在Spring 2.0中添加了session(和request)会话范围的bean的支持.这个方法通过org.springframework.aop.target.scope.ScopedProxyFactoryBean 创建了一个session会话范围的bean的CGLIB或者JDK动态代理,并设置scopeMap为org.springframework.web.context.scope.SessionScopeMap.

因为jar是向后兼容的,因为只需要简单地构建Spring并替换WebWork带来的jar.(当你读到这里的时候Spring 2.0 M1可能已经发布了).

有两种方法来设置这个,也就是使用或者不使用简化的XML.第一个方法是使用传统的bean定义,这对于理解内部发生了什么是很有用的.

一个shopping cart例子里经过改良的applicationContext.xml,使用了传统的XML DTD的例子显示在下面.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">

<beans>
    <bean id="shoppingCart" class="org.springframework.aop.target.scope.ScopedProxyFactoryBean"
            singleton="false">
        <property name="scopeKey" value="shoppingCart"/>
        <property name="targetBeanName" value="__shoppingCart"/>
        <property name="scopeMap">
            <bean class="org.springframework.web.context.scope.SessionScopeMap"/>
        </property>
    </bean>

    <bean id="__shoppingCart" class="com.opensymphony.webwork.example.ajax.cart.DefaultCart"
            singleton="false"/>

</beans>
```

一个使用简化XML方式的shopping cart例子的改良applicationContext.xml显示在下面.

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:aop="http://www.springframework.org/schema/aop"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
       http://www.springframework.org/schema/aop
http://www.springframework.org/schema/aop/spring-aop.xsd">

    <bean id="catalog" class="com.opensymphony.webwork.example.ajax.catalog.TestCatalog"
          singleton="true"/>

    <bean id="shoppingCart" class="com.opensymphony.webwork.example.ajax.cart.DefaultCart"
          singleton="true">
        <aop:scope type="session"/>
    </bean>

</beans>
```

你还要修改web.xml来包含下面的filter.

```
<filter>
    <filter-name>springFilter</filter-name>
    <filter-class>org.springframework.web.filter.RequestContextFilter</filter-class>
</filter>
<filter-mapping>
    <filter-name>springFilter</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

使用一个ServletFilter来定制 TargetSource

一个很"clean"的一般是为web应用程序提出的解决方案可以在 JA-SIG 发现. 这个解决方案有很好的文档,可以在 这里 发现. 在后面你可以找到一个采用它的方法.

XWork/WebWork 特殊的解决方案

前言

WebWork是基于XWork的,而XWork不是绑定到web层的.因此当处理session会话范围的对象时,WW用户可能会使用XWork的session抽象特性来保持他们的应用程序独立于web环境.这也就是我们为什么要讨论一些下面列出的XW/WW特定的解决方案.

定制 TargetSource, WebWork 的方法

这是一个上面指出的TargetSource解决方案的一个改良版本,它和已有的WebWork session相结合,不需要一个额外的filter或者listener.使用方法是几乎相似的,为你的对象创建一个接口,并确保你使用的总是接口而不是具体的实现,否则自动装配会失败.你可以研究 WebWorkTargetSource Shopping Cart Example 以找到更多的关于这个如何工作的信息.

```
package org.tuxbot.webwork.spring;

/* Portions Copyright 2005 The JA-SIG Collaborative.  All rights reserved.
 *  See license distributed with this file and
 *  available online at http://www.uportal.org/license.html
 */
```

```
import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;
import org.springframework.aop.target.AbstractPrototypeBasedTargetSource;
import org.springframework.beans.factory.DisposableBean;
import com.opensymphony.xwork.ActionContext;

import java.util.Map;

/**
 * This target source is to be used in collaberation with WebWork.
 * The target source binds the target bean to the Session retrieved from
 * WebWork. By default the bean is bound to the session
 * using the name of the target bean as part of the key. This can be overridden by setting
 * the <code>sessionKey</code> property to a not null value.
 *
 * @author Eric Dalquist <a href="mailto:edalquist@unicon.net">edalquist@unicon.net</a>
 * @author Eric Molitor <a href="mailto:eric@tuxbot.com">eric@tuxbot.com</a>
 * @version 1.0
 */
public class WebWorkTargetSource extends AbstractPrototypeBasedTargetSource implements
DisposableBean {
    private final static Log LOG = LogFactory.getLog(WebWorkTargetSource.class);

    private transient Object noSessionInstance = null;
    private String sessionKey = null;
    private String compiledSessionKey = null;

    public WebWorkTargetSource() {
        this.updateBeanKey();
    }

    /**
     * @return Returns the sessionKey.
     */
    public String getSessionKey() {
        return this.sessionKey;
    }
    /**
     * @param sessionKey The sessionKey to set.
     */
    public void setSessionKey(String sessionKey) {
        this.sessionKey = sessionKey;
        this.updateBeanKey();
    }
    /**
     * @see
org.springframework.aop.target.AbstractBeanFactoryBasedTargetSource#setTargetBeanName(java.lang.String)
     */
    public void setTargetBeanName(String targetBeanName) {
        super.setTargetBeanName(targetBeanName);
        this.updateBeanKey();
    }

    /**
     * @see org.springframework.aop.TargetSource#getTarget()
     */
    public Object getTarget() throws Exception {
        final Map session = ActionContext.getContext().getSession();

        if (session == null) {
            LOG.warn("No Session found for thread '" + Thread.currentThread().getName() + "'");

            if (this.noSessionInstance == null) {
                this.noSessionInstance = this.newPrototypeInstance();

                if (LOG.isDebugEnabled()) {
                    LOG.debug("Created instance of '" + this.getTargetBeanName() + "', not
bound to any webWorkSession.");
                }
            }
            else {
                if (LOG.isDebugEnabled()) {
                    LOG.debug("Found instance of '" + this.getTargetBeanName() + "', not bound
to any webWorkSession.");
                }
            }
            return this.noSessionInstance;
        }
        else {
            String beanKey = this.compiledSessionKey;

            Object instance = session.get(beanKey);
            if (instance == null) {
                instance = this.newPrototypeInstance();
                session.put(beanKey, instance);

                if (LOG.isDebugEnabled()) {
                    LOG.debug("Created instance of '" + this.getTargetBeanName() + "', bound to
webWorkSession for '"
                           + Thread.currentThread().getName() + "' using key '" + beanKey + "'.");
                }
            }
            else if (LOG.isDebugEnabled()) {
                LOG.debug("Found instance of '" + this.getTargetBeanName() + "', bound to
webWorkSession for '"
                       + Thread.currentThread().getName() + "' using key '" + beanKey + "'.");
            }

            return instance;
        }
    }

    /**
     * @see org.springframework.beans.factory.DisposableBean#destroy()
     */
    public void destroy() throws Exception {
        if (this.noSessionInstance != null && this.noSessionInstance instanceof DisposableBean)
{
            if (LOG.isDebugEnabled()) {
                LOG.debug("Destroying sessionless bean instance '" + this.noSessionInstance +
"'");
            }

            ((DisposableBean)this.noSessionInstance).destroy();
        }
    }

    /**
     * Generates the key to store the bean in the session with.
     */
    private void updateBeanKey() {
        if (this.sessionKey == null) {
            final StringBuffer buff = new StringBuffer();

            buff.append(this.getClass().getName());
            buff.append("_");
            buff.append(this.getTargetBeanName());

            this.compiledSessionKey = buff.toString();
        }
        else {
            this.compiledSessionKey = this.sessionKey;
        }
    }
}
```

自定义 ApplicationContext 实现

TODO: Document

自定义 WW/XW ObjectFactory

TODO: Document

Session 支持的 Bean Factory

这个想法就是简单地创建一个 获取/创建(session) 的bean factory:

```
package net.itneering.core.spring.session;

import org.springframework.beans.BeansException;
import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.BeanFactoryAware;

import com.opensymphony.xwork.ActionContext;

import java.util.Map;
import java.io.Serializable;

/**
 * SessionBackedBeanFactory tries to lookup beans by name in XWork session. If not found,
 * it tries to instantiate new bean and attaches it to said session.
 *
 * @author <a href="mailto:gielen@it-neering.net">Rene Gielen</a>
 */

public class SessionBackedBeanFactory implements Serializable, BeanFactoryAware {

    BeanFactory beanFactory = null;

    /**
     * Find a component by name in session scoped storage implementation. If not found, try to
instantiate new one by
     * {@link org.springframework.beans.factory.BeanFactory#getBean(String)}. Then found
component will be attached
     * to session store implementation.
     *
     * @param componentName
     * @return The requested component, if found.
     */
    public Object getSessionComponent( String componentName ) {
        Object result = getSession().get(componentName);
        if ( result == null ) {
            result = beanFactory.getBean(componentName);
            storeComponent(componentName, result);
        }
        return result;
    }

    public void storeComponent(String componentName, Object component ) {
        getSession().put(componentName, component);
    }

    /**
     * Actual implementation of the session scoped storage Map.
     * Lookup {@link com.opensymphony.xwork.ActionContext#getSession()}.
     *
     * @return The Map for keeping session objects.
     */
    public Map getSession() {
        return ActionContext.getContext().getSession();
    }

    /**
     * Callback that supplies the owning factory to a bean instance.
     * <p>Invoked after population of normal bean properties but before an init
     * callback like InitializingBean's afterPropertiesSet or a custom init-method.
     *
     * @param beanFactory owning BeanFactory (may not be null).
     *                    The bean can immediately call methods on the factory.
     *
     * @throws org.springframework.beans.BeansException
     *          in case of initialization errors
     * @see org.springframework.beans.factory.BeanInitializationException
     */
    public void setBeanFactory( BeanFactory beanFactory ) throws BeansException {
        this.beanFactory = beanFactory;
    }
}
```

示例的applicationContext 设置(注意session会话范围的bean设置为singleton="false"):

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans default-autowire="byName">
    <bean id="sessionBeanProxy"
class="net.itneering.core.spring.session.SessionBackedBeanFactory" singleton="true"/>
    <bean id="securityContextComponent"
class="net.itneering.security.component.DefaultSecurityContextComponent" singleton="false" />
</beans>
```

使用它的例子action:

```
package net.itneering.xwork.action;

import com.opensymphony.xwork.ActionSupport;
import net.itneering.core.spring.session.SessionBackedBeanFactory;
import net.itneering.security.component.DefaultSecurityContextComponent;

/**
 * Simple sessionBeanProxy aware action.
 *
 * @author <a href="mailto:gielen@it-neering.net">Rene Gielen</a>
 * @version $Revision: 1.1 $
 */

public class SecurityAwareAction extends ActionSupport implements PrincipalAware {

    private static final Logger log = Logger.getLogger(SecurityAwareAction.class);

    protected SessionBackedBeanFactory sessionBeanProxy;

    /**
     * For Spring wiring usage.
     *
     * @param sessionBeanProxy The sessionBeanProxyto use.
     */
    public void setSessionBeanProxy( SessionBackedBeanFactory sessionBeanProxy ) {
        this.sessionBeanProxy = sessionBeanProxy;
    }

    /**
     * Getter for actions security context.
     *
     * @return The securityContextComponent set by IoC.
     */
    public SecurityContextComponent getSecurityContextComponent() {
        return sessionBeanProxy!= null ?
sessionBeanProxy.getSessionComponent("securityContextComponent") : null;
    }

    /**
     * Get the current User Principal for this session.
     *
     * @return The User Principal.
     */
    public UserEntity getPrincipal() {
        try {
            return getSecurityContextComponent().getPrincipal();
        } catch ( NullPointerException e ) {
            return null;
        }
    }
    ...
}
```

对于使用广泛的session会话范围组件,你可能为了方便而集成SessionBackedBeanFactory:

```
package net.itneering.security.component;

import net.itneering.core.spring.session.SessionBackedBeanFactory;

/**
 * SecurityAwareSessionBeanProxy.
 *
 * @author <a href="mailto:gielen@it-neering.net">Rene Gielen</a>
 */

public class SecurityAwareSessionBeanProxy extends SessionBackedBeanFactory {

    String securityContextComponentName = "securityContextComponent";

    /**
     * Make component name configurable by spring setup
     */
    public void setSecurityContextComponentName( String securityContextComponentName ) {
        this.securityContextComponentName = securityContextComponentName;
    }

    public SecurityContextComponent getSecurityContextComponent() {
        return (SecurityContextComponent) getSessionComponent(securityContextComponentName);
    }

}
```

同样示例的 applicationContext 配置:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans default-autowire="byName">
    <bean id="mySecurityContextComponent"
class="net.itneering.security.component.DefaultSecurityContextComponent" singleton="false" />
    <bean id="sessionBeanProxy"
class="net.itneering.security.component.SecurityAwareSessionBeanProxy" singleton="true">
        <property name="securityContextComponentName" value="mySecurityContextComponent" />
    </bean>
</beans>
```

使用的action示例:

```
package net.itneering.xwork.action;

import com.opensymphony.xwork.ActionSupport;
import net.itneering.security.component.SecurityAwareSessionBeanProxy;
import net.itneering.security.component.DefaultSecurityContextComponent;

/**
 * Simple sessionBeanProxy aware action.
 *
 * @author <a href="mailto:gielen@it-neering.net">Rene Gielen</a>
 * @version $Revision: 1.1 $
 */

public class SecurityAwareAction extends ActionSupport implements PrincipalAware {

    private static final Logger log = Logger.getLogger(SecurityAwareAction.class);

    protected SecurityAwareSessionBeanProxy sessionBeanProxy;

    /**
     * For Spring wiring usage.
     *
     * @param sessionBeanProxy The sessionBeanProxy to use.
     */
    public void setSessionBeanProxy( SecurityAwareSessionBeanProxy sessionBeanProxy ) {
        this.sessionBeanProxy = sessionBeanProxy;
    }
```

```
        /**
         * Get the current User Principal for this session.
         *
         * @return The User Principal.
         */
        public UserEntity getPrincipal() {
            try {
                return sessionBeanProxy.getSecurityContextComponent().getPrincipal();
            } catch ( NullPointerException e ) {
                return null;
            }
        }
        ...
    }
```

就像前面所说,这个解决方案是非常简单的.你不需要绑定到web层,配置也实在简单,也没有必要在applicationContext.xml里面定义proxy.

主要的缺点就是你不能直接装配session会话范围的bean到你的action,你必须间接使用session支持的bean factory.而且,当你处理XWork session抽象层的时候,你经常必须关心来设置一个action context.

自动代理session支持的组件Factory

有人有这样一个实现吗? (Eric Molitor)

这个意图有一点不同,因此我试图澄清标题.好主意,虽然... (Rene Gielen).

译注:下面这段就不翻译了

The theory here is to create a custom Pointcut class that utilizes the ComponentConfiguration retrieved from the DefaultComponentManager (which loads the Component list from components.xml). The getClassFilter() matches anything that implements one of the Components in the ComponentConfiguration. The Pointcut is then registered as an advisor for all beans (AutoProxy via Springs DefaultAdvisorAutoProxyCreator). The Advice implementation looks at which Component is implmented and fetches the apporiate value out of the Session and calls the Components setter method.

TODO: Document, create example

This page last changed on Apr 24, 2006 by scud.

WebWorkTargetSource Shopping Cart 例子

这是一个shopping cart例子的改造版本,使用了WebWorkTargetSource.它是一个快速的修改来展示如何使用WebWorkTargetSource,并不是一个完整的解决方案或者可以使用的模板.如果我的文档不清楚(可能)或者没有一个有意义(也十分可能),只需要用这些版本的替换已存在的 DefaultCart.java 和 applicationContext.xml ,然后运行例子即可.

DefaultCart 修改

为了让自动装配可以工作两处对DefaultCart.java的修改是需要的.当spring去寻找bean来自动装配它时,会看到两个ShoppingCarts而不知所措,因为让自动装配工作的话就需要仅看到一个.为了避免这个问题需要修改DefaultCart不再实现ShoppingCart接口.虽然有一个fun的内部类和内部接口让这变得有点复杂.为了让DefaultCart能编译(而且能工作)需要修改所有对CartEntry的引用需要改为ShoppingCart.CartEntry.

```
package com.opensymphony.webwork.example.ajax.cart;

import com.opensymphony.webwork.example.ajax.catalog.Product;

import java.util.HashMap;
import java.util.HashSet;
import java.util.Map;
import java.util.Set;

import sun.reflect.Reflection;

/**
 * DefaultCart - Poorly Modified by Eric Molitor <eric@tuxbot.com>
 *
 * @author Jason Carreira <jcarreira@eplus.com>
 */
public class DefaultCart {
    Map contents = new HashMap();

    public static DefaultCart getCart() {
        return new DefaultCart();
    }

    public void addToCart(int quantity, Product product) {
        ShoppingCart.CartEntry entry = (ShoppingCart.CartEntry) contents.get(product);
        if (entry == null) {
            entry = new DefaultCartEntry(quantity, product);
            contents.put(product, entry);
        } else {
            entry.setQuantity(entry.getQuantity() + quantity);
        }
    }

    public void setQuantity(int quantity, Product product) {
        if (quantity <= 0) {
            contents.remove(product);
            return;
        }
        ShoppingCart.CartEntry entry = (ShoppingCart.CartEntry ) contents.get(product);
        if (entry == null) {
            entry = new DefaultCartEntry(quantity, product);
            contents.put(product, entry);
        } else {
            entry.setQuantity(quantity+entry.getQuantity());
        }
    }

    public void removeFromCart(Product product) {
        contents.remove(product);
    }

    public Set getContents() {
        return new HashSet(contents.values());
    }

    public int getQuantityForProduct(Product product) {
        ShoppingCart.CartEntry entry = (ShoppingCart.CartEntry) contents.get(product);
        if (entry == null) {
            return 0;
        }
        return entry.getQuantity();
    }

    public String toString() {
        return "DefaultCart{" +
                "contents=" + contents +
                "}";
    }

    public static Object getBean() {

       System.out.println("!!!!!!!!!!!! Parent is:" + Reflection.getCallerClass(1));
        new Exception("poo").printStackTrace();
       return new DefaultCart();
    }

    class DefaultCartEntry implements ShoppingCart.CartEntry {
        private int quantity;
        private Product product;

        public DefaultCartEntry(int quantity, Product product) {
            this.quantity = quantity;
            this.product = product;
        }

        public int getQuantity() {
            return quantity;
        }

        public void setQuantity(int quantity) {
            this.quantity = quantity;
        }

        public Product getProduct() {
            return product;
        }
    }
}
```

applicationContext.xml 改动

为了得到一个session特性的shopping cart,我们需要修改actionContext来调用我们的WebWorkTargetSource.我们通过一个ProxyFactory来做到这一点,它创建一个一个基于我们的接口(ShoppingCart)的对象,并使用targetSource来调用我们定制的TargetSource(WebWorkTargetSource).为了获取并创建新的实例,WebWorkTargetSource因此也需要知道底层的实现.我们传递一个引用到新的shoppingCartTarget bean的定义,这个定义仅仅引用了我们新的DefaultCart.为了保持事情不被搞乱,我们设置两个bean都是使用by name的自动装配.

```
    <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
    "http://www.springframework.org/dtd/spring-beans.dtd">

    <beans default-autowire="autodetect">

        <bean id="catalog"
              singleton="true"
              class="com.opensymphony.webwork.example.ajax.catalog.TestCatalog"/>

        <bean id="shoppingCart" class="org.springframework.aop.framework.ProxyFactoryBean"
```

```
autowire="byName">

        <property name="proxyTargetClass">
            <value>true</value>
        </property>
        <property name="proxyInterfaces">
            <list>
                <value>com.opensymphony.webwork.example.ajax.cart.ShoppingCart</value>
            </list>
        </property>
        <property name="targetSource">
            <bean class="org.tuxbot.webwork.spring.WebWorkTargetSource">
                <property name="targetBeanName">
                    <value>shoppingCartTarget</value>
                </property>
            </bean>
        </property>
    </bean>

    <bean id="shoppingCartTarget"
          class="com.opensymphony.webwork.example.ajax.cart.DefaultCart"
          singleton="false" autowire="byName">
    </bean>

</beans>
```

This page last changed on Mar 30, 2006 by scud.

Using Tiles with WebWork

This page is under construction. Thank you for your patience.

Per blogpost by Matt Raible, the idea was launched to integrate Tiles with WebWork.

Several approaches are possible:

- Use the TilesResult
- Use Tiles configured by Spring (example)

Action Chaining

This page last changed on Apr 30, 2006 by scud.

Overview

WebWork提供把多个Action按照预先定义好的顺序或者流程链接起来的能力.这个特性通过给指定的Actions设置一个Chain Result,然后通过一个ChainingInterceptor 拦截目标Action来实现.

警告

通常不推荐使用Action链.但是,你可以有其他选择,例如redirect after post技术

Chain Result

Chain Result是一种result 类型,它基于自己的拦截器stack(堆栈)和result调用一个action,这样允许一个action附带着原来的状态将请求转到目标action,下面是关于如何定义这样的一个序列的例子.

```
<package name="public" extends="webwork-default">
    <!-- Chain creatAccount to login, using the default parameter -->
    <action name="createAccount" class="...">
        <result type="chain">login</result>
    </action>

    <action name="login" class="...">
        <!-- Chain to another namespace -->
        <result type="chain">
            <param name="actionName">dashboard</param>
            <param name="namespace">/secure</param>
        </result>
    </action>
</package>

<package name="secure" extends="webwork-default" namespace="/secure">
    <action name="dashboard" class="...">
        <result>dashboard.jsp</result>
    </action>
</package>
```

在这个Action(参见 配置文件)执行之后同一命名空间(或者是默认的""命名空间)的另外一个Action也会得到执行.一个可选的"namespace"参数来指定另外一个命名空间的Action

Chaining Interceptor

如果你需要把Action链中早先的Actions的属性复制到当前的Action中, 你应该采用ChainingInterceptor. 这个拦截器会从request和ValueStack中复制所有的原始参数传给目标Action. ValueStack会记录Action源的信息,这样目标Action就可以通过ValueStack访问早先的Action的属性,这些属性也能被Action链中最后的result访问,例如JSP或者Velocity页面.

Action链通常用来提供查找列表(例如状态的下拉列表等等).既然这些Actions已经放入了ValueStack,它们的属性在View层就是可用的.这个功能也可以使用Action标签(ActionTag)在显示页面执行一个Action来做到.你也可以利用Redirect Action Result来完成这样的功能.

This page last changed on May 12, 2006 by scud.

ActionMapper

ActionMapper用来负责在HTTP请求和action调用请求之间进行一个映射,反之亦然.当提供了一个 HttpServletRequest,如果没有action调用请求映射,ActionMapper可能会返回null,否则会可能返回一个描述了一个action调用的ActionMapping,WebWork会试着去调用.ActionMapper没必要保证返回的ActionMapping是一个真正的action,也就是说保证是一个合法的请求.这意味这大多数的ActionMapper不需要考虑WebWork的配置来确定一个请求是否被映射了.

就像一个请求可以从HTTP映射到一个action调用,相反的映射也是可以的.当然,因为HTTP请求(当在HTTP回应中显示时)必须是字符串的格式,一个字符串被返回,而不是一个实际的请求对象.

DefaultActionMapper

缺省情况下,DefaultActionMapper 会被使用:
缺省的action mapper实现,使用的是标准的 *.[ext] (这里 ext(后缀) 通常是 "action") 模式.这个后缀从WebWork的配置主键webwork.action.extension中获取.

为了帮助处理按钮和其他相关的需求,这个mapper(我们希望,其他ActionMapper也能这样)具有这样的功能:使用一些预定义的前缀命名一个按钮,这些按钮会引发执行行为.这四个前缀是:

- Method 前缀- method:default
- Action 前缀- action:dashboard
- Redirect 前缀- redirect:cancel.jsp
- Redirect-action 前缀- redirect-action:cancel

除了这四个前缀,这个mapper也明白 foo!bar的action命名方式,或者扩展方式(例如 foo!bar.action) ,或者前缀方式(例如 action:foo!bar).这个语法告诉mapper映射到名称为foo的action和对应的方法bar.

Method 前缀

使用method前缀,来代替调用baz action的 execute()方法(缺省是这个,如果在xwork.xml里面没有被设置成其他方法的话),baz action的 anotherMethod会被调用.一个非常优雅的方法来确定那个按钮被点击了.作为选择,当点击时,submit按钮可以在action上设置一个特殊的值(one would have submit button set a particular value on the action when clicked),execute()方法依赖那个按钮被点击来决定如何处理设置的值.

```
<ww:form name="baz">
    <ww:textfield label="Enter your name" name="person.name"/>
    <ww:submit value="Create person"/>
    <ww:submit name="method:anotherMethod" value="Cancel"/>
</ww:form>
```

Action 前缀

使用action前缀,代替执行baz action的execute()方法(缺省是这个,如果在xwork.xml里面没有被设置成其他方法的话),anotherAction action的execute()方法(再次假定在xwork.xml里面它没有被重置)会被执行.

```
<ww:form name="baz">
    <ww:textfield label="Enter your name" name="person.name"/>
    <ww:submit value="Create person"/>
    <ww:submit name="action:anotherAction" value="Cancel"/>
</ww:form>
```

Redirect 前缀

使用redirect前缀,代替执行baz action的execute()方法(缺省是这个,如果在xwork.xml里面没有被设置成其他方法的话),它会进行转向,在这里是 www.google.com. 在内部,它使用ServletRedirectResult 来完成这个任务.

```
<ww:form name="baz">
    <ww:textfield label="Enter your name" name="person.name"/>
    <ww:submit value="Create person"/>
    <ww:submit name="redirect:www.google.com" value="Cancel"/>
</ww:form>
```

Redirect-action prefix

使用redirect-action前缀,代替执行baz action的execute()方法(缺省是这个,如果在xwork.xml里面没有被设置成其他方法的话),它会转向,在这里是 'dashboard.action'.在内部,它使用ServletRedirectResult 来执行这个任务,并从webwork.properties里读取后缀.

```
<ww:form name="baz">
    <ww:textfield label="Enter your name" name="person.name"/>
    <ww:submit value="Create person"/>
    <ww:submit name="redirect-action:dashboard" value="Cancel"/>
</ww:form>
```

ActionMapperFactory

你可以定义你自己的ActionMapper来配置ActionMapperFactory:

工厂创建了ActionMapper.这个工厂查找在WebWork的配置文件里*webwork.mapper.class*主键定义的ActionMapper类名.

使用自己的ActionMapper可以定义你自己的清晰的命名空间,例如URL类似 /person/1 ,就类似于使用DefaultActionMapper时对应的请求 /getPerson.action?personID=1 .

App Servers

This page last changed on May 08, 2006 by dogg.

以下列表所列出的应用服务器WebWork都能在其中工作, and any potential workarounds necessary for full functionality:

- WebLogic
- WebLogic 6.1
- SunOne 7.0
- WebSphere
- JRun
- Jetty
- Tomcat/JBoss
- Resin
- Orion
- OC4J

SunOne 7.0

This page last changed on May 08, 2006 by dogg.

你需要授于WebWork许可权限:

```
grant {
        permission java.security.AllPermission;
};
```

或者更明确,

- 授予写入权限 java.util.PropertyPermission.
- 添加 java.lang.reflect.ReflectPermission "suppressAccessChecks"
- OgnlInvoke Permission

```
grant {
        permission java.util.PropertyPermission "*", "read, write";
        permission java.lang.reflect.ReflectPermission "suppressAccessChecks";
        permission ognl.OgnlInvokePermission "*";
};
```

WebLogic

This page last changed on Mar 30, 2006 by scud.

```
> The classloaders of WLS seem not to play nice with velocity when
> deploying this way.
>
> If you haven't already tried, do the following, it makes it all work
> for us:
>
> 1) In the webwork.properties file (which should be in your
> WEB-INF/classes directory) put a line like this:
>
> webwork.velocity.configfile = my-velocity.properties
>
> 2) create a "my-velocity.properties" file under WEB-INF/classes and
> put into it the contents of the velocity.properties file that is in
> webwork's velocity-dep.jar
>
> 3) in your new "my-velocity.properties" file, find the section titled
> "T E M P L A T E L O A D E R S", and change this section to look like
> this:
>
> ===========================================
> resource.loader = class
>
> file.resource.loader.description = Velocity File Resource Loader
> file.resource.loader.class =
> org.apache.velocity.runtime.resource.loader.FileResourceLoader
> file.resource.loader.path = .
> file.resource.loader.cache = false
> file.resource.loader.modificationCheckInterval = 2
>
> class.resource.loader.class =
> org.apache.velocity.runtime.resource.loader.ClasspathResourceLoader
> class.resource.loader.cache = true
> ===========================================
>
> ... which straightens out the class resource loading problems (for us
> at least).
>
> hope this helps,
> james
```

Note: The "when deploying this way" comment above refers to deploying a war file (not expanded) into the deployment directory of WebLogic; with WL 8.x, this is typically at <bea_home>/user_projects/domains/mydomain/.

This page last changed on Mar 30, 2006 by scud.

Running WebWork 2 on Weblogic Server 6.1

This document describes why WebWork 2 doesn't work "as-is" on Weblogic Server 6.1 and shows how to build an additional JAR that will fix the problems.

Note: the service pack of Weblogic Server 6.1 used is SP4.

The first part of this document describes the technical problems and the theoretical solution.

Why WebWork Doesn't Work

Weblogic 6.1 was published just prior to the finalization of the Servlet 2.3 specification. The incompatibility is that servlet filters and listeners in Weblogic 6.1 do not work with the 2.3 spec primarily because the servlet context is not retrieved in the same way. This causes virtually all filter initialization operations to fail with an AbstractMethodError exception.

How WebWork Is Modified

In Servlet 2.3, the servlet context is available from the session object; this is not true for Weblogic Server 6.1. Hence, filters and listeners must be modified to retrieve the servlet context from a different source; this is accomplished by retrieving the servlet context from the FilterConfig passed to the servlet filters during initialzation.

However, the WebWork code cannot be modifed to do this, because this will break the Servlet 2.3 specification. The goal is to leave the "original" WebWork modified so that it is still Servlet 2.3 compatible, and then to add an additional JAR that "breaks" WebWork to work on Weblogic Server 6.1.

Hence, if you want to run WebWork under Servlet 2.3, the default, then simply build WebWork as usual.

But if you want to run WebWork under Servlet 2.3, you need to build the additional JAR and put it into your WAR file, and then modify your web.xml to use the new classes instead of the standard ones.

The standard WebWork has already been modified slightly to make the above effort possible:

1. RequestLifecycleFilter is modified to retrieve its servlet context from the method getServletContext(). This method, getServletContext(), is then implemented to return the servlet context from where it is available in Servlet 2.3: the session object. The logical operation is unchanged, but now subclasses can override getServletContext() to retrieve the servlet context from a different location as we'll see below.
2. SessionLifecycleListener is modified in the same way as RequestLifecycleFilter. The method, getServletContext(), is implemented to return the servlet context, in this case also from the session object. Again, subclasses can override the getServletContext() method to restore the servlet context from a different source. Again, this class's functionality is unchanged.

Now, in a separate project, the following classes are added and compiled into a separate JAR:

RequestLifecycleFilterCompatWeblogic61

This subclass of RequestLifecycleFilter simply overrides getServletContext() to retrieve the servlet context from the filter config, creates a singleton class, SessionContextSingleton, and assigns the servlet context to the singleton so that the listeners will have the ability to retrieve it.

SessionLifecycleListenerCompatWeblogic61

This subclass of SessionLifecycleListener simply overrides getServletContext() to retrieve the servlet context from the singleton created above.

FilterDispatcherCompatWeblogic61

Although the superclass of this class, FilterDispatcher, is commented out, this subclass retrieves the servlet context in the same way as RequestLifecycleFilterCompatWeblogic61 in case it is ever resurrected. At this time, this class is unnecessary.

ServletContextSingleton

A singleton class whose sole purpose is to hold the servlet context so that listener classes have access to it.

Setting Up WebWork 2 to Run on Weblogic 6.1

Building Your Own Project

In the web.xml file, make the following class name substitutions:

Old Class Name	New Class Name
RequestLifecycleFilter	RequestLifecycleFilterCompatWeblogic61
SessionLifecycleListener	SessionLifecycleListenerCompatWeblogic61
FilterDispatcher	FilterDispatcherCompatWeblogic61

FAQ

I still get the AbstractMethodError Exception when Weblogic Server starts up. What am I doing wrong?

1. Check to see if a webwork-example.war is still lingering in your mydomain/applications folder and delete it if it is there.
2. See next FAQ question.

The server behavior seems like it is from a previous source code base; I can't debug it. What's the clue?

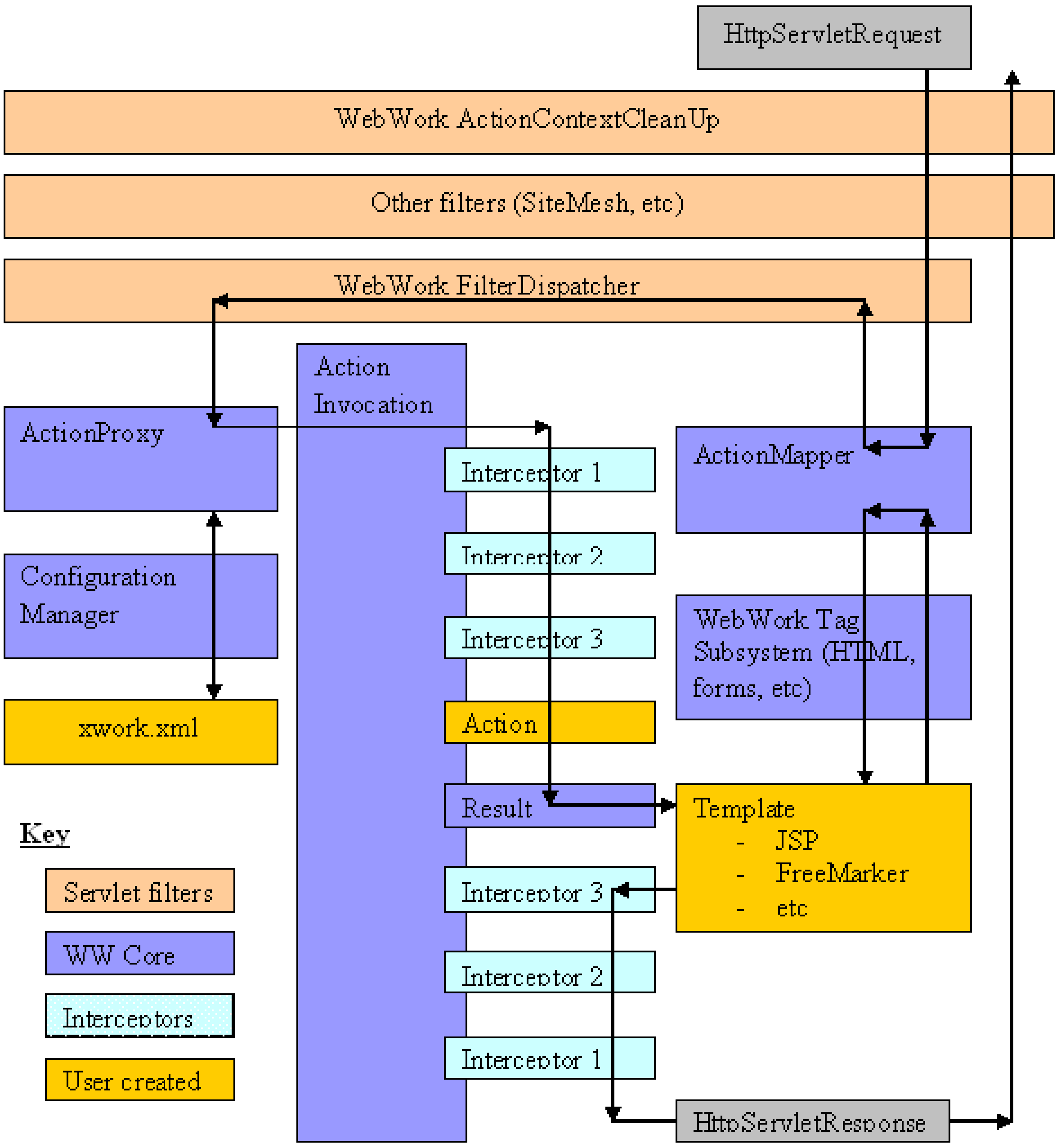
Sometimes BEA Weblogic Server doesn't "rebuild" its temporary files. Do the following to force the temporary files to rebuild:

1. Stop the server.
2. Delete the .wlnotdelete folder in mydomain/applications.
3. Restart the server.

Architecture

This page last changed on May 18, 2006 by scud.

WebWork的架构最好用一张图来表示:



如此图所示,一个初始的请求被发送到Servlet容器(如Tomcat或Resin),这个请求经过一个标准的Filter链,其中包括(可选的)ActionContextCleanUp Filter,如果你要在应用程序中整合其他的技术如SiteMesh,就需要使用这个这个Filter.然后请求经过 FilterDispatcher(译者注:WW2.2之前处理这个请求的是ServletDispatcher),在它里面ActionMapper会判断这个请求是否需要调用Action.

如果ActionMapper决定应该调用一个Action,FilterDispatcher就把请求委托给ActionProxy,ActionProxy通过WebWork的配置文件管理器读取xwork.xml文件里的配置信息.然后创建一个实现了命令模式的ActionInvocation.这一过程包括在调用action本身之前调用所有的interceptor(before()方法)

一旦action方法返回,ActionInvocation就要查找xwork.xml文件中这个action的结果码(action result code)(译者注:一个String如success,input)所对应的result,然后执行这个result.通常情况下result会调用JSP或FreeMarker模版来呈现页面(但不总是这样,例如result也可以是一个Action链).当呈现页面时,模版可以使用WebWork提供的一些标签.其中一些组件可以和ActionMapper一起工作来为后面的请求呈现恰当的URL.

⚠ 在这个架构中的所有对象(action,result,interceptor等等)都是通过ObjectFactory创建的.ObjectFactory是可插入,因此可以和Spring,Pico这样的框架整合的很好.如果需要,Webwork中的对象也可以由你自己提供的ObjectFactory来创建.

最后interceptor被再次执行(顺序和开始相反,调用after方法),然后最终请求被返回给web.xml中配置的其他Filter.如果已经设置了ActionContextCleanUp Filter,那么FilterDispatcher就不会清理ThreadLocal中的ActionContext信息.如果没有设置ActionContextCleanUp Filter,FilterDispatcher会清理掉所有的ThreadLocal.

Articles and press

This page last changed on Apr 25, 2006 by scud.

WebWork是一个很流行的框架和社区.同样地,有很多文章,演讲稿,关于WebWork的书籍,这里列出一些例子.

书籍

- WebWork in Action - Patrick Lightbody, Jason Carreira; Manning; September 2005
- Java Open Source Programming - Joseph Walnes, Ara Abrahamian, Mike Cannon-Brookes, Patrick Lightbody; Wily; November 2003
- Art of Java Web Development - Neal Ford; Manning; November 2003
- WebWork Live - Matthew Porter; SourceBeat; N/A

演讲稿

- WebWork in Action: A hands on look at the future of Struts (slides) - Patrick Lightbody; Portland Java Users Group, February 2005
- WebWork + AJAX: A winning combination (video, slides) - Patrick Lightbody; JavaZone; August 2005
- WebWork in Action: An introduction to WebWork (video, slides) - Patrick Lightbody; JavaZone; August 2005
- WebWork 2.0: Strutting the OpenSymphony way - Jason Carreira; TheServerSide Symposium; April 2004
- Strutting the OpenSymphony way - Mike Cannon-Brookes; TheServerSide Symposium; July 2003
- WebWork 2.0 Overview - Rick Salsa; Groove Systems

文章

- Interview about the Struts merger - Interview of Patrick Lightbody
- Validation with WebWork 2.2 - Zarar Siddiqi
- Working wit the WebWork Framework - Vlad Kofman
- Building with WebWork - Kris Thompson; TheServerSide; November 2003
- Tutorial and article in Brazilian Portuguese - January, 2004

博客

官方博客可以在 here. 看到

除此之外,WebWork开发者的博客可能会提供一些有用的信息:

- Blogbody - Patrick Lightbody
- Jason Carreira

博客文章:

- A summary of what is new in WebWork 2.0 - Mike Cannnon-Brookes

Strutting the OpenSymphony way

This page last changed on Mar 30, 2006 by scud.

WebWork: Strutting the OpenSymphony way - Mike Cannon-Brookes

- An overview of Mike's thoughts from the conference
- Mike's original PPT

WebWork is a pull-based MVC framework focused on componentization and code reuse. It is currently in beta, but is being used by several opensource projects and a few commercial projects in development. This is the second generation of WebWork, which was originally developed by Rickard Oberg, and in this release, what was WebWork has been broken into two projects, [XW:XWork] and WebWork.

XWork is a generic command pattern implementation with absolutely NO ties to the web. XWork provides many core services, including interceptors, meta-data based validation, type conversion, a very powerful expression language (OGNL - the Object Graph Notation Language) and an Inversion of Control (IoC) container implementation.

WebWork provides a layer on top of XWork to do HTTP request / response handling. It includes a ServletDispatcher to turn HTTP requests into calls to an Action, session and application scope mapping, request parameter mapping, view integration with various web view technologies (JSP, Velocity, FreeMarker, JasperReports), and user interface components in the form of JSP tags and Velocity macros wrapped around reusable UI components.

An Action is the basic unit of work in WebWork. It is a simple command object that implements the Action Interface, which has only one method: execute(). Action implementers can extend the ActionSupport class, which provides i18n localization of messages (with one ResourceBundle per Action class and searching up the inheritance tree) and error message handling including class level and field level messages.

Actions can be developed in one of two styles: Model driven or field driven. Model driven Actions expose a model class via a get method, and the form fields refer directly to the model properties using expressions like "pet.name". XWork uses Ognl (the Object Graph Notation Language) as its expression language, and when rendering the page, this expression will translate to getPet().getName(). When setting properties, this will translate to getPet().setName(). This style of development allows for a great deal of model reuse and can allow you to directly edit your domain objects in your web pages, rather than needing a translation layer to form beans. Field driven Actions have their own properties which are used in the view. The action's execute() method collates the properties and interacts with the model. This can be very useful when your form and model are not parallel. Even in this case, the powerful expression language in WebWork can allow you to compose your form fields into aggregate beans, such as an address bean, which you can reuse to simplify your action classes.

WebWork allows you to build your own reusable UI components by simply defining a Velocity template. This is how the pre-built components of WebWork are built for common components such as text fields, buttons, forms, etc. and made available from any view type (either JSP or Velocity at the moment). These components are skinnable by defining multiple templates for the same component in different paths. If your components include the default header and footer templates that are used in the pre-built templates, then they will inherit the ability to automatically handle displaying error messages beside the problem form field. These custom UI components are especially handy for reusing templates which handle your custom model types or for things like date pickers, which Mike showed as an example.

Interceptors in XWork allow common code to be applied around (before and/or after) action execution. This is what Mike calls "Practical AOP". Interceptors help to decouple and componentized your code. Interceptors can be organized into stacks, which are lists of interceptors to be applied in sequence, and can be applied to actions or whole packages. Much of the core functionality of XWork and WebWork is implemented as Interceptors. The common basic examples of Interceptors are timing and logging, and these are built in with XWork. Mike went through an example of an interceptor to identify users of events via email. This interceptor has its own external configuration file which specifies which users are interested in which events, and it compares this configuration with the action invocations passing through it to determine if any messages should be sent.

XWork's validation framework allows for decoupled validation of action properties. It is implemented as an Interceptor and reads external XML files which define the validations to be applied to the Action. Error messages are loaded from the Action's localized messages and flow through to the UI. Validator classes can be plugged in to add to the set of bundled validators. The bundled validators include required field and required String validators, range validators for Dates and numbers, and email and URL validators. XWork also includes expression validators at both the Action and field level which allow you to use any Ognl expression as the validation.

Inversion of Control (IoC) removes the burden of managing components from your code and puts it on the container. The container takes care of managing component lifecycle and dependencies. EJB is an example of IoC, but with limited services. IoC promotes simplicity and decoupling of your components and encourages your classes to be smaller and more focused. Unit testing is also simplified, as you can just supply MockObject instances of the services your code depends upon during testing. XWork and WebWork provide a web-native IoC container which manages component dependencies. In WebWork IoC is implemented as lifecycle managers (SessionLifecycleListener, etc) and an Interceptor. There are 4 component scopes in WebWork IoC: Application, HTTP Session, HTTP Request, and Action invocation. IoC in XWork / WebWork is purely optional, so you can use it if you want it.

XWork / WebWork allows for sets of Actions and their views to be bundled as a jar file and reused. Your main xwork.xml file can include the xml configuration file of the jar file because they are included from the classpath. Similarly, if your views are Velocity templates, you can bundle your views in the jar file and they will be loaded from the classpath when rendering. This allows for componentization of your application and reuse of bundled Actions across applications.

I have to admit, when Mike mentioned this feature, I thought he was crazy. I didn't say anything at the session, but asked him about it later, and he said "didn't you write the package include stuff?" I'll take it as a good sign that things can be used in a different way than was imagined.

Mike finished up with a comparison of WebWork vs. Struts . Struts is obviously the 500 lb gorilla in the web MVC space, so why use WebWork? WebWork's pros include being a smaller, simpler framework, not having to build ActionForm beans, making it very simple to test your Actions, having multiple well-supported view technologies, simpler views with less JSP tags and a more powerful expression language, not having to make your Actions thread-safe, not having your Actions tied to the web, and not being part of Jakarta . WebWork also adds many new features such as Interceptors, packages, IoC, etc. WebWork's cons include being a smaller project with fewer books and less tool support, having less standards support for specs like JSTL and JSF, and not being part of Jakarta .

Building WebWork

This page last changed on Mar 30, 2006 by scud.

Why Building?

In most cases you will not need to build WebWork yourself, since the distribution package contains all you need to get started and productive with WebWork. See Getting Started for more information on how to start working with the distributed binaries. However, there are situations where you want to build WebWork from scratch, for example if you want to try out tweaks and patches, or simply if want to check the head of current development. For the latter, a solution apart from building WebWork from scratch might be to have a look into our Ivy Repository containing continous integration builds ("nightly builds"), containing the latest build of WebWork and XWork jars.

Getting the Sources

Distribution

The current distribution packages of WebWork contain all sources, as well as all needed libraries for building jars and running. Distribution packages are found here.
The dependency resolution via Ivy is disabled by default for the build from a distribution package (> Webwork 2.2 Beta 4). If you need to work with the Clover and Ivy-related buildfile tasks, you might want to follow the instructions below.

CVS

The sources are are hosted via CVS on java.net. So getting your sources is quite standard:

1. If you have not already done, login to repository:
   cvs -d :pserver:guest@cvs.dev.java.net:/cvs login
2. Checkout the the webwork sourcetree:
   cvs -d :pserver:guest@cvs.dev.java.net:/cvs checkout webwork
3. Checkout the the opensymphony common sourcetree:
   cvs -d :pserver:guest@cvs.dev.java.net:/cvs checkout opensymphony/common

If you are a registered user at Java.net, you might use your username instead of anonymous guest account. For detailed information on how to setup different clients, visit https://webwork.dev.java.net/servlets/ProjectSource.

Building

We assume that you are familiar with ant as the standard build tool in the Java world.

What is Ivy?

If you checked out the sources from CVS, you might have noticed that the lib directory is empty. Unfortunately this does not mean that webwork has no external dependecies at all. To be honest, as a full featured MVC framework it has lots of dependencies, which in turn means that there has to be some dependency management. This is where Ivy comes to play.
Ivy is a free java based dependency manager, with powerful features such as transitive dependencies, maven repository compatibility, continuous integration, html reports and many more. Ivy is fully integrated with ant, so you do not have to get into a complicated tool. See http://jayasoft.org/ivy for details.

Installing and using Ivy

The installation is quite trivial: Put a copy of the ivy-1.x.jar found in the common directory of the opensymphony module in your $ANT_HOME/lib directory.
If you want to test the Ivy functionality, ensure you have an internet connection. Change into the webwork module directory and execute ant init(as you might guess, any other task depends on init). Ivy will now resolv all dependencies and (hopefully) download all required jars and put it into the lib directory.
See Dependencies for informations on how to integrate Ivy in your own Webwork2 based projects.

Skipping dependency resolution (> Webwork 2.2 Beta 4)

The build now knows the property "skip.ivy". This may be specified from build.properties file or from a command line ant execution with -Dskip.ivy=true. If set, dependency resolution via Ivy is omitted and build is done with current jars found in lib directory.
This behaviour is turned on by default for builds from the distribution package.

JUnit and Clover

The full build process will require JUnit and Clover.
Place a copy of junit.jar (>= 3.8.1) and clover.jar (>= 1.3.9) into your $ANT_HOME/lib directory (if not already exists). If you don't have these jars at hand, look in the lib/build directory of your WebWork module after you called ant init in the step before...
OpenSymphony Clover license is found in the common directory of the opensymphony module. Place the clover-license.jar into your $ANT_HOME/lib directory as well. Now you are ready to ...

Build

Call ant jar or simply ant to build the WebWork jars. Play around with other targets, as you like.

JDK/JRE Compatibility

WebWork requires JDK 1.4.2+ to build. JDK 5.0 is not required for building.
WebWork-based applications require JRE 1.4.2+ to run. JRE 5.0 is not required to run unless your application uses the optional xwork-tiger module, which adds some Java 5.0 specific features to XWork/WebWork functionality.

Chat

This page last changed on Mar 30, 2006 by scud.

WebWork Jabber Chat

WebWork has an official chat room on a jabber server, where some of the developers hang out and some important discussions take place. In contrast to the official meetings, which are mostly scheduled, you can always drop by to ask questions or say hi.

Register

In order to connect to the chat server, you need to make an account on the Opensymphony Forums. You'll need that username and password to connect.

Watch uppercase/lowercase ! If you have a username with a capital in it, you'll need to make a second account with an all lowercase username (due to a small bug in the chat server)

Clients

If you log in at the forum, you can join the chat using a ajax client (click 'join!' next to the **Chatting Now**). This is very useful if you don't have a real jabber client at your disposal.

- Chat server: conference.chat.opensymphony.com
- Chat room: webwork-users
- Username: your username from the forum
- Password: your password from the forum
- Handle (optional): your nickname

Meetings Minutes

This page last changed on Mar 30, 2006 by scud.

This is a central location for the chat logs and minutes for all WebWork-related meetings.

- 07\-04\-2005 Documentation
- 06\-15\-2005 Documentation
- 06\-01\-2005 AJAX

06-01-2005 AJAX

This page last changed on Mar 30, 2006 by scud.

Transcript

```
   PSquad33        my biggest concerns are making sure we don't bite off more than we can chew
and that we keep WebWork... WebWork
        jcarreira       Have you looked at Cameron's stuff?
        iroughley       what were your thoughts on Camerons checkins?
        PSquad33        i'm not interested in becoming a vehicle for distributing dojo
        iroughley       yes
        jcarreira       I wanted to talk to him about contributing his stuff back to dojo
        PSquad33        all the javascript widget stuff sounds like it is getting a little too
far away from webwork. but still, it is interesting
        PSquad33        anyway, i'll be on and off
        iroughley       I like dojo as the transport, but as soon as you get too far into JS I
am concerned abouot browser compatibility
        jcarreira       I like the idea of making the components dojo widgets, but I want the
JSP tags to write out the dojo widgets... that way we can maintain server-side state
        jcarreira       yeah, that's a good reason to put it in dojo... more people using it =
more compatibility testing
        iroughley       exactly
        PSquad33        are we going to support dojo entirely, or will this be optional stuff?
i just don't want to divert focus
        PSquad33        maybe we should make an seperate project or something. do my concerns
make sense?
        jcarreira       well, I think we should have an Ajax theme
        iroughley       I like using it as the transport, and the event stuff is good.
        jcarreira       I don't know... I think it's important to have rich widgets as part of
the framework...
        PSquad33        sure -- but we can't abandon our base either
        jcarreira       ...and they're going to do drag-n-drop, etc.
        PSquad33        people still use the plain old XHTML theme
        PSquad33        i want to keep the goals in sync
        PSquad33        anyway, my big concern is that I feel a bit left out of the loop -- and
if I do, our users defintiely will
        iroughley       I guess my underlying issue with the code that cameron added, was that
it had no dependency with webwork
        PSquad33        so you guys need to really hold my hand on this if you think this is
the right direction to go
        PSquad33        show me the vision in simple to understand terms
        PSquad33        ian: exactly
        PSquad33        ian: that is exactly my concern -- we're going to fragment the user
base at this rate
        jcarreira       yeah.. I don't want to get into the biz of maintaining add-ons to dojo
        iroughley       it could be used with or without webwork, and then we are developing
for someone else
        jcarreira       I wanted to talk to him about contributing the JS stuff back to dojo,
and we just use it
        jcarreira       I've been trying to think through if we need the server-side events too
        iroughley       so then move in the direction that I think we are going - tag libs
around the attributes for the dojo elements
        jcarreira       yep
        iroughley       so that we can link into server side state
        jcarreira       Should we just maintain UI widget dependencies on the client-side?
        jcarreira       or should we try to implement a real MVC style where the model changes
cause the correct view components to update?
        iroughley       this is a urious question
        iroughley       i like loose dependencies on the client, via topics
        iroughley       so far, for what I am doing it works well
        jcarreira       so the UI components just know to listen to topics and refresh
themselves?
        iroughley       yes
        jcarreira       in that case it can just be a UI component theme
        iroughley       the caveat is that the components have a known grnularity, that is
understood by the developer
        iroughley       yes
        jcarreira       except for the extra JSP attributes
        iroughley       and changed per theme
        jcarreira       what do you mean about the granularity?
        iroughley       you have to know that a remote div (for example) might get a list from
```

```
the server, or render a domain object
        jcarreira       I was thinking the remote div would just get HTML and set its innerHtml
        iroughley       so when you call that div to refresh itself you are doing work to
obtain the information
        jcarreira       that way it's just like a ww:action tag...
        iroughley       yes
        iroughley       the action does the work
        jcarreira       and if you have it like this:
        jcarreira       <ww:remotediv action="foo"/>
        jcarreira       actually...
        jcarreira       this could just be a theme for the ww:action tag...
        jcarreira       ...except it's not a UI tag
        iroughley       true
        jcarreira       but it could wrap a ww:action tag and delegate calls to it
        jcarreira       but maybe that's limiting its reusability
        jcarreira       <ww:remotediv href="foo.action"><ww:action name="foo"/></ww:remotediv>
        jcarreira       the contents of the ww:remotediv would be the original content (no need
for a separate call to load the content), then the href would be used when it refreshes
        iroughley       that's a good idea
        iroughley       i haven't had a chance to implement the initial content yet (as per
your suggestion last week)
        jcarreira       what are the params to the remotediv tag?
        iroughley       so, i guess that the UI doesn't need to maintain dependencies on the
client, just make calls via remote div's (or other components) and let the actions determine
the state
        jcarreira       yeah
        jcarreira       but we need things like select boxes which can populate
        |<-- PSquad33 has left efnet (Read error: Connection reset by peer)
        iroughley       is, url, updateFreq, loadingText, reloadingText, errortext,
showErrorTransportText, topicName
        jcarreira       for example, if I'm shopping and I select a catalog, it should populate
the product categories in a select list automatically
        jcarreira       ...I'd also like to be able to have a lazy-loading tree widget
        iroughley       i agree
        jcarreira       the loadingText can go away.. just make it the body enclosed by the tag
        iroughley       that was my plan
        jcarreira       we need a way to tell it to load immediately... then the default text
would be there while it loads the first time
        iroughley       :)
        iroughley       my idea exactly
        iroughley       monday I hope it will be done
        jcarreira       cool
        iroughley       could the select tag and tree tag use topics to refresh state from the
server
        jcarreira       what do you mean?
        iroughley       then use DWR or dojo to obtain JS to refresh themselves
        jcarreira       ah
        -->| PSquad33 (~PSquad32@c-67-160-147-93.hsd1.or.comcast.net) has joined #webwork
        PSquad33        back
        PSquad33        sorry about that
        PSquad33        you guys still talking about stuff
        jcarreira       but where do they call to get the stuff to refresh themselves?
        PSquad33        ?
        jcarreira       yep
        iroughley       yeah
        jcarreira       you need to bind something to DWR to populate the list, right?
        iroughley       the data has to be sourced from somewhere originally, right?
        iroughley       yep
        jcarreira       yeah... originally it might be in the action
        iroughley       exactly
        PSquad33        let me know when you guys are ready to talk strategy. i don't have much
to add technically, but i want to make sure we are absolutely clear on our strategy and market
perception.
        jcarreira       would we want to populate the data on the client?
        iroughley       what do you mean?
        jcarreira       pre-populate a data structure and select from there...
        |<-- PSquad33 has left efnet (Remote host closed the connection)
        iroughley       personally I wouldn't - if you are using ajax the assumption is that
the data may change, and should be refreshed from the server
        iroughley       that doesn;t mean that we shouldn't have a static tree widget
        jcarreira       I guess I don't know how DWR works... how do you tell it what to call?
Can it load data from the session?
        iroughley       DWR (i think) makes a call to a method on a serverside object - like a
spring BO
        jcarreira       can it access the session to get its data?
        iroughley       i would think that you could have it access a WW action, and return the
session information from there
```

```
        iroughley       i haven't looked at it lately
        jcarreira       I'm asking Patrick... he keeps getting kicked off
        iroughley       do you want to try another IRC server?
        jcarreira       nah, it's his client
        jcarreira       he's got a cheesy shareware one for mac
        iroughley       ok - so if anyone changes the select list they update an element in the
session - then the select list uses DWR to call an action with a list name (?) which obtains a
refreshed (possibly) list from the session
        jcarreira       I was using an onChange() call... it uses bind() to call and set
something into the session
        iroughley       that would work
        jcarreira       we need a way to make it easy for users to bind onChange to a call to
the server
        jcarreira       You shouldn't have to write a JS function everytime
        -->| PSquad32_ (~chatzilla@c-67-160-147-93.hsd1.or.comcast.net) has joined #webwork
        PSquad32_       yo yo
        PSquad32_       chatzilla to the rescue
        jcarreira       ok, Patrick was just explaining DWR to me
        iroughley       but it's more complex than that, as it might not be the same element -
it may be a different one
        iroughley       ok
        jcarreira       no, it shouldn't change anything client-side
        PSquad32_       so yeah, lemme know when you are ready to talk strategy. dojo doesn't
seem to line up with what webwork is about at all, so i'm a bit worried
        jcarreira       it should make a call to the server, set some session state, then
return and send a client-side topic message
        iroughley       ok - yes
        jcarreira       it's maybe a little more chatty, but it's simple to understand and
model
        jcarreira       and very loosely coupled
        iroughley       could we set up each of the UI tags to define whether they send or
update, and then wire it up for the users?
        iroughley       more chatty, but the calls are going to be very small
        jcarreira       what do you mean?
        jcarreira       I think each UI component should be able to be given a URL to hit with
the new value onChange()
        jcarreira       and a topic to publish to
        iroughley       agreed
        iroughley       so that call will be small (not alot of data in the URL)
        jcarreira       and also have a topic to listen to that would load the value... so that
would need a url to get the list
        iroughley       yes
        iroughley       and that call is going to be returning just the list, and not a
complete HTML page - do the data being sent back to the client will be small
        jcarreira       yep.. but in what form does it return the list?
        jcarreira       ...patrick, we're almost done with the Js stuff....
        iroughley       innerHTML?
        jcarreira       what does it mean if you set the innerHtml of a select tag? what do you
set it to?
        iroughley       it's easy and whatever is called on the server to get the data out of
the session can format it to html
        iroughley       I would think that any tag can be gotten by id, and then html set in it
        iroughley       so <OPTION>one</OPTION> etc.
        PSquad32_       i'm going to play a game of halo -- be back in a few :)
        jcarreira       yes, but what do you set into the innerHtml of a select tag to populate
it? is it just the <options>?
        iroughley       i think so
        iroughley       haven't tried it, but it makes sense
        jcarreira       ok, so for the tree... it needs to get javascript back, probably, or
some data structure... should we support both?
        Fracture        Hi
        jcarreira       Hi Fracture...
        Fracture        i'm going to read the log to catch up
        iroughley       i think that depends on the core tree widget that we use and augment
for the implementation
        jcarreira       true... I've been playing with dtree for a while... it's pretty easy
        jcarreira       Fracture, I'm not sure who you are IRL...
        iroughley       do you think that returning JS or HTML would be easier with dtree?
        jcarreira       JS
        iroughley       i think that's fine then - the action/object that obtains the data from
the session can modify it to JS to return to the browser
        jcarreira       This is how I add things to the tree:
        jcarreira       <ww:iterator value="categories">
        jcarreira       tree.add(<ww:property value="id"/>,<ww:property value="(parentCategory
== null)?0:parentCategory.id"/>,'<ww:property
value="name"/>','javascript:changeCategory(<ww:property value="id"/>);','<ww:property
value="name"/>','_top','/img/folder.gif','/img/folderopen.gif');
```

```
        jcarreira       </ww:iterator>
        jcarreira       then you add it to the page like this:
        jcarreira       document.write(tree);
        iroughley       cool
        iroughley       we could even use JS for the select - just an implementation detail, as
it will be hidden from teh end user
        jcarreira       but I think we can get to that later... I'm ok hardwiring this one
together for now.. but we should make it easy to create components that send changes to the
server to update server state and then send update events on topics
        Fracture        is Cameron :)
        jcarreira       ahh.. ok...
        =-=     Fracture is now known as Cameron-
        jcarreira       Cameron, let us know when you're caught up
        Cameron-        shall do .. half way there.
        iroughley       i would agree with that
        iroughley       while we are handwiring, we should also start thinking about a naming
convention for the topics
        jcarreira       categoryChange, userChange, etc?
        iroughley       yes - I have been using topic_{html id}_event - i.e.
topic_myRemoteDivId_refresh
        iroughley       but they can easily change
        jcarreira       I think we should make it easy to type and remember... users will end
up needing to do some wiring themselves at some point
        iroughley       makes sense
        iroughley       on some of the elements i added a topic to publish to - perhaps I
should go ahead and add this to them all
        jcarreira       where would we be defining topics?
        iroughley       at the moment I autowire some, and give the user the option of
specifying them on some
        jcarreira       well, we should talk through that with Patrick... how we're going to
position this Ajax stuff... if we're going to add attributes to the taglibs, etc.
        jcarreira       which are autowired?
        jcarreira       brb
        iroughley       from memory I think the autowired ones are - remotediv sends a
"refesh", tab header sends a "tab selected"
        iroughley       remote div can specify a topic to listen to and refrsh itself - i think
that might be the only one for now
        Cameron-        ok - I agree that these dojo widgets that I have created really belong
in dojo - or a separate project. I was just demonstrating that rather than us write javascript
code (that the ww components use) we write them as dojo widgets, and use the components to bind
data to them. When using more interactive client controls - webwork more becomes a controller
for marshaling data. I do think that we need to bundle a dojo build with w
        iroughley       oh - the remote link / <a href> is autowired to send a "refresh"
        Cameron-        and also allow our users to use a different profile build - if they
meet our dependencies
        PSquad32_       cameron: this whole dependency on dojo widgets is very un-webwork-ish
        PSquad32_       i _really_ think it is not the right direction
        Cameron-        we depend on DRW, we depend on XmlHttpRequest..
        Cameron-        we have to have dependencies
        PSquad32_       but we're depending on a framework
        PSquad32_       the others are libraries
        PSquad32_       dojo is a framework
        PSquad32_       think about it from user's perspective. when they upgrade to 2.2, what
are they going to see?
        PSquad32_       what is the "webwork message"?
        PSquad32_       what is our story? we can't have 4 different versions of the same story
        jcarreira       well, that's where we need to decide if this is a theme
        jcarreira       I think this is the Ajax theme
        jcarreira       and it requires dojo
        jcarreira       I think we bundle a dojo profile with the example app
        PSquad32_       i think that is fine (a lot of this is marketing more than anything)
        iroughley       are there cases where you might needs to utilize multiple themes for a
single compnent?
        jcarreira       yes
        iroughley       i.e. 2 different styles of lists or tabbed controls on the same page
        iroughley       so each of these would then need to be developed from teh core ajax
theme
        jcarreira       sometimes I end up using the "simple" theme because the layout requires
I can't put it in the table
        jcarreira       what do you mean?
        iroughley       it's late - i think i just confused myself :)
        jcarreira       what's our status with JSP 2?
        jcarreira       with JSP 2 you can just add more attributes without having to put them
in the TLD, right?
        PSquad32_       i am fine iwth requiring jsp 2
        jcarreira       I don't really know the features of JSP 2 :-)
        jcarreira       but requiring JSP 2 is requiring J2EE 1.4 and J2SE 1.4, right?
        jcarreira       which means no WebSphere 4.x or 5.0
        PSquad32_       websphere 5 is JDK 1.4
        jcarreira       5.1 is
        iroughley       if that is the case i would opt not to exclude those markets
        PSquad32_       that's fine
        PSquad32_       we don't need JSP 2 features
        PSquad32_       ww:param works
        PSquad32_       i have to run
        PSquad32_       let's keep thinking about what the webwork story is -- how all this
fits in
        PSquad32_       we can't just say the old stuff doens't matter or isn't supported
        Cameron-        so Pat - what is the wework story ?
        PSquad32_       well, i think we all need to work on that
        PSquad32_       i'll draft up some thoughts -- maybe you guys can too
        jcarreira       ok, so if not JSP2, then we need to pull those attributes out and use
ww:param to parameterize the tag for this specific theme
        PSquad32_       my main concern is that you guys understand what i'm saying :)
        PSquad32_       anyway, i have to run
        PSquad32_       see ya
        jcarreira       later
        Cameron-        bye
        iroughley       later
        jcarreira       so if this is going to be a theme, we can't add theme-specific
attributes
        iroughley       for the remote div that cameron added - right?
        Cameron-        you can - using ww:param no ?
        jcarreira       right, I mean attributes to the TLD
        Cameron-        you could probably add optional ones, and document them as 'only used
for xxx theme'
        jcarreira       Hmm... could be.... as long as it's very clear... I don't want to be
answering questions about why the topic isn't being used with the xhtml theme
        Cameron-        true
        jcarreira       speaking of which, does anyone know if someone actually build a real
XHTML theme using div's and CSS instead of tables?
        jcarreira       built
        iroughley       no idea
        Cameron-        no idea
        iroughley       a while back i thought cloves and pat were talking about doing it
        Cameron-        yeah - I rememer hearing that.
        iroughley       so the plan is to move forward with a dojo/ajax theme?
        jcarreira       yes
        jcarreira       who wants to take what?
        Cameron-        with regards to your 'topics' ... are they for use client side .. or do
these events get routed to an action too ?
        jcarreira       Cameron, you're going to talk to the dojo guys about contributing the
remote div, remote link, and remote submit stuff to dojo?
        jcarreira       they're the dojo.event.Topic stuff that lets you do publish/subscribe
in JS
        Cameron-        yep - i'll do that.
        iroughley       we will also require theme specific tags then, right?
        Cameron-        so you use topics for inter widget communication ?
        iroughley       remote div for example
        jcarreira       I built it as a wrapper around their event stuff
        iroughley       yes - mainly for server side refreshes at the moment
        jcarreira       I think let's keep the theme specific attributes for now
        jcarreira       the idea is for a component to update its server state then publish a
message on a topic
        Cameron-        so something emits an event, a widget is triggered, which then
refreshes itself ?
        Cameron-        sure
        jcarreira       yep, the developer specifies with the tags which topic(s) a component
listens to to refresh itself
        iroughley       or parts of itself - like the drop down list elements
        jcarreira       and which topic a component publishes on when it is changed
        jcarreira       so they are loosely coupled and the developer specifies which thing
listens to the others
        Cameron-        cool.
        jcarreira       so about dojo and source
        iroughley       so - there are 3 parts as i see it - 1. the dojo UI widgets, 2. the
events/topics, and 3. the WW tag/component that combines them all
        Cameron-        yep
        iroughley       go ahead jason
        jcarreira       should we just bundle the __package__.js?
        Cameron-        we also need the other 'non bundled' files.. like html templates, css
files and other widget resources (gif/jpg)
        jcarreira       yeah, the tags generate JS to use the dojo widgets and JS to wire them
to topics
```

```
        jcarreira       ok
        jcarreira       where do those come from?
        Cameron-        I setup an ant target dojo-profile that causes dojo to build a
__package__.js and copy it and other non bundled resources into the os/static/dojo package
        Cameron-        where do those come from ? dunno what you mean
        jcarreira       Well, I don't know about bundling the sources to dojo into webwork's
CVS
        jcarreira       I think we should just bundle the final built product, like we do with
other libraries
        iroughley       also, if they are in webwork.static, can the end user override the
images easily if they wanted?
        Cameron-        that depends on how the widet is authored
        Cameron-        authored
        jcarreira       how does one specify the images to be used?
        jcarreira       can it be done so that the template could be edited to change where
they come from?
        Cameron-        the widget can take a img= attribute and use it in the building of the
rendered widget.
        iroughley       ok - so it doesn't need to be in ww.static
        jcarreira       well, the defaults can be
        Cameron-        doesn't have to be.. however .. defaults may be packaged there
        iroughley       ok - makes sense
        jcarreira       ok, so the order of things:
        jcarreira       1) Cameron, you're going to try to get this stuff into dojo
        jcarreira       2) we build a profiles of dojo with the widgets
        jcarreira       3) we edit the JSP templates to use the dojo widgets
        jcarreira       sound good?
        jcarreira       anything i'm missing?
        iroughley       sounds good
        Cameron-        with the remote div.. why do we need a jsp template ?
        jcarreira       because we want to wire it to topics
        Cameron-        why not have a <dojo:topic> widget ?
        jcarreira       and it can take the body contents as the default content of the div
        Cameron-        the body content can be done <dojo:remotediv attribs...><ww:action
name='foo'/></dojo:remotediv>
        jcarreira       what would the dojo:topic do?
        Cameron-        the dojo:topic would 'install' the topic 'bus' into the page
        jcarreira       well, I'd like to keep things consistent... using ww: tags and
auto-wiring them to topics
        iroughley       can th edojo:topic tag specify the topic to listen to and the topic to
publish to when an event ocurrs?
        iroughley       it's also going to be more code for developers to type
        jcarreira       <ww:remotediv listenTopic="itemSelected"> creates JS to wire the div to
refresh when it gets a message on that topic
        jcarreira       yeah, we want to make this dead-simple
        Cameron-        dojo:remotediv listenTopic='itemSelected'> can do the same thing..
without a jsp template
        jcarreira       ...and it doesn't hurt to wrap dojo in case something else comes along
and we replace the JS library
        Cameron-        I think we only need to use a jsp tag when we need to access action
context from the tag
        jcarreira       I don't want to make users know anything about dojo
        Cameron-        ok.. well. I think we have two different approaches.
        iroughley       if we don't wrap it, it won't really be a theme - wil it?
        jcarreira       yeah, the JSP template can just call the dojo stuff, so you can just
use the dojo stuff directly, too
        iroughley       it will be something else to use with ww
        Cameron-        ok
        jcarreira       well... maybe we should just make this a ww:div tag
        iroughley       makes sense
        jcarreira       and in the Ajax theme it can become a remote div
        iroughley       then we also need a <ww:a>
        iroughley       :)
        jcarreira       yeah... we should do that
        jcarreira       it can use the URLTag code
        Cameron-        sounds good
        jcarreira       ok, Ian, do you want to take those?
        iroughley       sure
        jcarreira       cool
        iroughley       we also need dojo components for the existing elements - remote div,
remote a link, tabbed panel - right?
        jcarreira       yeah... not sure how hard it is to do the tabbed panel... that's a more
complicated one
        iroughley       tonight we also talked about a select list and tree element
        iroughley       it might be easy - just use <ww:div> and it will just not update -
events will be the only issue
        jcarreira       ...and I still want them to build a nice menuing system :-)
        iroughley       the only other thing I was thinking about is a paginated list
        jcarreira       let's get the ones we've got now using the new architecture
        jcarreira       JSP tag -> template -> dojo widget
        iroughley       cameron - you up for building the dojo components?
        Cameron-        yeah
        jcarreira       ok, I'll fill Patrick in with where we're heading and maybe we can
schedule another chat to talk about the WebWork message, etc
        Cameron-        ok
        iroughley       ok - then I will start on the remote div tag/template from teh dojo
component next monday
        Cameron-        we need a wiki page to list the ajax theme components
        Cameron-        (so I know what to build)
        iroughley       let me know as new ones come down the pipe
        iroughley       the current tutorial list all the ones i've done -
localhost/webwork/tutorial/ajax/...
```

07-04-2005 Documentation

This page last changed on Mar 30, 2006 by scud.

Attendees

- Vitor Souza
- Jay Bose
- Alexandru

Outcome

Look to this updated page: http://wiki.opensymphony.com/pages/viewpage.action?pageId=4795

Transcript

```
[7/4/2005 5:17 PM] <jaybose> forget it, i am.
[7/4/2005 5:17 PM] <the_mind> what do you mean by recording?
[7/4/2005 5:17 PM] <jaybose> logging
[7/4/2005 5:17 PM] <jaybose> so others can view it lat4er
[7/4/2005 5:18 PM] <the_mind> i thought i can copy and paste it :">
[7/4/2005 5:18 PM] <jaybose> you can do that too
[7/4/2005 5:18 PM] <jaybose> :-)
[7/4/2005 5:18 PM] <jaybose> hey, did you read his page?
[7/4/2005 5:18 PM] <the_mind> sorry to ask: who are you nightfal?
[7/4/2005 5:18 PM] <the_mind> whose page?
[7/4/2005 5:20 PM] <jaybose> Vitor made a page:
http://wiki.opensymphony.com/pages/viewpage.action?pageId=4795
[7/4/2005 5:20 PM] <jaybose> that's what we'll be working off of
[7/4/2005 5:20 PM] <jaybose> the main purpose of this meeting is to nail down a TOC
[7/4/2005 5:20 PM] <the_mind> yep... i just wanted to see if anything else comes out
[7/4/2005 5:20 PM] <jaybose> and then get the dev team in general to clear it
[7/4/2005 5:21 PM] <jaybose> dev team really meaning Patrick and Jason
[7/4/2005 5:21 PM] -->| vitor (c91d085c@chat.efnet.org) has joined #webwork
[7/4/2005 5:21 PM] <jaybose> hey Vitor
[7/4/2005 5:21 PM] <jaybose> ready to go?
[7/4/2005 5:22 PM] <the_mind> from my pov yes (... almost falling asleep :-( )
[7/4/2005 5:22 PM] <jaybose> haha
[7/4/2005 5:22 PM] <vitor> Hey. Sorry, I'm struggling with this webchat.
[7/4/2005 5:22 PM] <jaybose> np
[7/4/2005 5:23 PM] -->| vitorsouz (~vircuser@201.29.8.92) has joined #webwork
[7/4/2005 5:23 PM] <vitorsouz> This is much better. Using vIRC now.
[7/4/2005 5:23 PM] |<-- vitor has left efnet (Remote host closed the connection)
[7/4/2005 5:23 PM] <vitorsouz> Don't know what happened. mIRC couldn't connect. I even got a
D-Lined message! ouch!
[7/4/2005 5:23 PM] <jaybose> hmm
[7/4/2005 5:23 PM] <vitorsouz> Sorry for this big mess... Ready to go now.
[7/4/2005 5:24 PM] <jaybose> ok, cool
[7/4/2005 5:24 PM] <jaybose> umm forst off
[7/4/2005 5:24 PM] <jaybose> looking at your page
[7/4/2005 5:24 PM] <jaybose> we want an overview
[7/4/2005 5:24 PM] <vitorsouz>  Ok. Let me open it here too
[7/4/2005 5:25 PM] <vitorsouz> Meaning you want me to explain what I'm thinking?
[7/4/2005 5:25 PM] <jaybose> kind of, what should the overview way about WW?
[7/4/2005 5:25 PM] <jaybose> say*
[7/4/2005 5:26 PM] <jaybose> I noted what WW is, and what it is not
[7/4/2005 5:26 PM] <vitorsouz> Ok.
[7/4/2005 5:26 PM] <jaybose> so that would mean a MVS web framework built on XWork
[7/4/2005 5:26 PM] <jaybose> maybe say waht XWork is brielfy
[7/4/2005 5:26 PM] <jaybose> MVC*
[7/4/2005 5:27 PM] <vitorsouz> Good.
[7/4/2005 5:28 PM] <vitorsouz> I think the overview should briefly describe the software and
```

```
bring other information that's interesting to people that are thinking of evaluating it.
[7/4/2005 5:28 PM] <jaybose> like?
[7/4/2005 5:28 PM] <the_mind> i think that articles and press and testimonials is a very good
way to introduce WW
[7/4/2005 5:28 PM] <vitorsouz> So I placed: what is WebWork (could include what it is not),
comparison to Struts and others, something about the community, articles and testimonials.
[7/4/2005 5:29 PM] <jaybose> ok
[7/4/2005 5:29 PM] <the_mind> a comparison to other solution cannot be very detailed so the
user may be lost already
[7/4/2005 5:29 PM] <vitorsouz> It should not teach WW to anyone. That's the reference and the
tutorial's jobs.
[7/4/2005 5:29 PM] <jaybose> great.
[7/4/2005 5:29 PM] <vitorsouz> themind: I think the reference is for experienced users only.
[7/4/2005 5:29 PM] <vitorsouz> I mean, comparison.
[7/4/2005 5:29 PM] <vitorsouz> The comparison is for experienced users. People that know
another framework.
[7/4/2005 5:30 PM] <the_mind> yep but you want this put into the overview
[7/4/2005 5:30 PM] <jaybose> evaluators will find a comparison helpful
[7/4/2005 5:30 PM] <vitorsouz> It's a section of the overview. I don't think the overview is
meant for sequential reading.
[7/4/2005 5:30 PM] <the_mind> if i am a new user i would like to read this from an independent
way... so pointing to articles
[7/4/2005 5:30 PM] <jaybose> and they will definitely look in the overview.
[7/4/2005 5:31 PM] <vitorsouz> We could call it an "Appendix" of the overview, to stress that
it's not essential for the newbie. But I'm not sure this is really needed.
[7/4/2005 5:32 PM] <the_mind> moreover spending time to do a full cycle comparison - when these
are already available is no use... just my 2c
[7/4/2005 5:32 PM] <vitorsouz> You're right about that: it should be mostly pointers to other
people's evaluations, eg. Matt Raible's.
[7/4/2005 5:32 PM] <jaybose> we already have one
[7/4/2005 5:32 PM] <jaybose> it
[7/4/2005 5:32 PM] <jaybose> it's a matter of placement
[7/4/2005 5:33 PM] <the_mind> excellent vitor
[7/4/2005 5:33 PM] <vitorsouz> That's another thing: the developers already wrote technical
differences between WW and Struts.
[7/4/2005 5:33 PM] <vitorsouz> So we would place that there and pointers to other articles.
[7/4/2005 5:33 PM] <the_mind> exactly
[7/4/2005 5:33 PM] <the_mind> or it can be a part of the FAQ
[7/4/2005 5:33 PM] <the_mind> i usually see this comparison included in FAQs
[7/4/2005 5:34 PM] <vitorsouz> Ok. Just so I'm not completely lost in the process here. Are we
starting from my suggestions? From Jay's suggestions? How is this discussion working?
[7/4/2005 5:34 PM] <the_mind> but this is not important for me (the placement)
[7/4/2005 5:34 PM] <jaybose> I am pulling things from your suggestions
[7/4/2005 5:34 PM] <vitorsouz> Okay.
[7/4/2005 5:34 PM] <the_mind> i was reading this
http://wiki.opensymphony.com/pages/viewpage.action?pageId=4795
[7/4/2005 5:35 PM] <the_mind> and commenting around - nothing more :-(
[7/4/2005 5:35 PM] <vitorsouz> That's alright. Sounds good to me. :)
[7/4/2005 5:35 PM] <jaybose> Ok, anymore on Overview, or move on to Project Information?
[7/4/2005 5:35 PM] <vitorsouz> That would be my question, exactly.
[7/4/2005 5:35 PM] <the_mind> from my pov i can move on
[7/4/2005 5:36 PM] <jaybose> ok so Prj Info, what do we want in here?
[7/4/2005 5:36 PM] <vitorsouz> Just to wrap up, then: I'll add "(What WW is not)" to the side
of "What is WebWork", just to emphasize.
[7/4/2005 5:36 PM] <the_mind> quite clear and complete imo
[7/4/2005 5:37 PM] <jaybose> ehhh
[7/4/2005 5:37 PM] <jaybose> i'm not for that
[7/4/2005 5:37 PM] <jaybose> anymore
[7/4/2005 5:37 PM] <the_mind> go on
[7/4/2005 5:37 PM] <the_mind> hit it
[7/4/2005 5:37 PM] <jaybose> let the reader figure it out
[7/4/2005 5:37 PM] <vitorsouz> Sorry. I'm lost again.
[7/4/2005 5:37 PM] <vitorsouz> You're not for "What WW is not"?
[7/4/2005 5:38 PM] <jaybose> by saying it's a web based MVC, that should be enough
[7/4/2005 5:38 PM] <jaybose> yeah.
[7/4/2005 5:38 PM] <vitorsouz> Ok.
[7/4/2005 5:38 PM] <the_mind> i would say about WW what is already said
[7/4/2005 5:38 PM] <vitorsouz> I'll add a note saying that.
[7/4/2005 5:38 PM] <the_mind> a MVC based on XWork (a command .... ) and that's it
[7/4/2005 5:39 PM] <jaybose> I'm taking a bunch of notes, i can add them to the page after the
meeting
[7/4/2005 5:39 PM] <vitorsouz> Ok. That's better than. I won't bother doing it too.
[7/4/2005 5:39 PM] <jaybose> Project Information: ...
[7/4/2005 5:39 PM] <vitorsouz> Ok. What would you guys add or remove?
[7/4/2005 5:39 PM] <jaybose> I'd keep.
[7/4/2005 5:40 PM] <the_mind> for me it is perfect... already said it
[7/4/2005 5:40 PM] <jaybose> I assume this is Team members, mailing lists, etc
```

```
[7/4/2005 5:40 PM] <vitorsouz> Mailing lists is included in "WebWork Community" under Overview.
[7/4/2005 5:40 PM] <the_mind> these are included already in overview
[7/4/2005 5:40 PM] <vitorsouz> Team is not explicit in the Project Information sections. You
could add it there later.
[7/4/2005 5:41 PM] <jaybose> so what did you think for that section?
[7/4/2005 5:41 PM] <the_mind> vitor i think it is a good idea to move everything related to ml,
team, etc to project information
[7/4/2005 5:41 PM] <vitorsouz> 1. License; 2. Deployment notes; 3. Versions; 4. Dependencies;
5. WebWork Team.
[7/4/2005 5:41 PM] <vitorsouz> Do you think it should be removed from Overview, then?
[7/4/2005 5:41 PM] <the_mind> and leave the Overview as a simple intro to the project
[7/4/2005 5:41 PM] <jaybose> I agree w/ the_mind, all that should leave Overview and go to Prj
Info
[7/4/2005 5:41 PM] <the_mind> yes that's it
[7/4/2005 5:41 PM] <vitorsouz> Ok. SO the whole "WebWork Community" section would move to
P.Info?
[7/4/2005 5:42 PM] <the_mind> oke with me +1
[7/4/2005 5:42 PM] <jaybose> yes
[7/4/2005 5:42 PM] <vitorsouz> Ok. I agree too. Then it's License, Deployment notes, Versions,
Dependencies, WebWork Team and Community.
[7/4/2005 5:43 PM] <vitorsouz> Not necessarily in this order.
[7/4/2005 5:43 PM] <jaybose> License, Deployment notes, Versions, Dependencies, WebWork Team,
Mailing Lists, Forum
[7/4/2005 5:43 PM] <the_mind> License, versions, dependencies, www and community, deployment
[7/4/2005 5:43 PM] <jaybose> When you say community, what does that currently mean?
[7/4/2005 5:43 PM] <the_mind> i see if license is good for me
[7/4/2005 5:44 PM] <the_mind> than i choose the version for which i see dependencies
[7/4/2005 5:44 PM] <the_mind> than i look for community stuff
[7/4/2005 5:44 PM] <the_mind> and if needed i will go to deployment tricks
[7/4/2005 5:44 PM] <vitorsouz> It's item number 3 under Overview: Mailing Lists/Forum, Bug
Tracker, Wiki, How to Contribute.
[7/4/2005 5:44 PM] <vitorsouz> The idea now is to move it to Prj. Info
[7/4/2005 5:44 PM] <the_mind> do not forget team members
[7/4/2005 5:45 PM] <jaybose> yep
[7/4/2005 5:45 PM] <the_mind> so let's see the order and move on
[7/4/2005 5:45 PM] <the_mind> :D
[7/4/2005 5:45 PM] <jaybose> ok, so Overview should be very light in comparison to what is it
now.
[7/4/2005 5:46 PM] <jaybose> We could hash out the order a little later.
[7/4/2005 5:46 PM] <vitorsouz> Yes. I'm updating the page so we close this issue...
[7/4/2005 5:46 PM] <the_mind> oke
[7/4/2005 5:46 PM] <vitorsouz> Ok. Refresh it and check if it's correct now.
[7/4/2005 5:46 PM] <the_mind> i will miss the important part :((
[7/4/2005 5:47 PM] <the_mind> good
[7/4/2005 5:47 PM] <jaybose> Overview looks great
[7/4/2005 5:47 PM] <jaybose> FAQ or more on Prj Info?
[7/4/2005 5:48 PM] <vitorsouz> I think FAQ could have it's own section because we could create
subsections, such as "Questions about the project", "Questions on Validators", "Questions on
Velocity/Freemarker", etc...
[7/4/2005 5:48 PM] <the_mind> tutorial or cookbook
[7/4/2005 5:48 PM] <vitorsouz> Hmmm... I got it now, Jay
[7/4/2005 5:48 PM] <vitorsouz> Nevermind what I said.
[7/4/2005 5:48 PM] <vitorsouz> I misunderstood your question :)
[7/4/2005 5:48 PM] <jaybose> yes, like Caucho does for Resin
[7/4/2005 5:48 PM] <jaybose> no you did not.
[7/4/2005 5:49 PM] <the_mind> i would let last the faq as passing through the other stuff it
will get more clear what should be in the faq
[7/4/2005 5:49 PM] <vitorsouz> I thought you meant putting FAQ under Prj. Info.
[7/4/2005 5:49 PM] <vitorsouz> Was that what you meant?
[7/4/2005 5:49 PM] <jaybose> haha, no. I agree on your structure idea
[7/4/2005 5:49 PM] <vitorsouz> Ok. So are we closed on Overview and Prj. Info?
[7/4/2005 5:49 PM] <jaybose> yep.
[7/4/2005 5:49 PM] <the_mind> yep
[7/4/2005 5:50 PM] <vitorsouz> All raise your hands. \o_ :P
[7/4/2005 5:50 PM] <vitorsouz> Just kidding. Moving on to FAQ, then.
[7/4/2005 5:50 PM] <the_mind> \^/
[7/4/2005 5:50 PM] <jaybose> :-)
[7/4/2005 5:50 PM] <jaybose> In terms of FAQ content, does WW have one now?
[7/4/2005 5:50 PM] <the_mind> i don't think so
[7/4/2005 5:50 PM] <vitorsouz> Not sure. Let me check.
[7/4/2005 5:50 PM] <the_mind> that's why i would let the faq be the last
[7/4/2005 5:51 PM] <jaybose> http://wiki.opensymphony.com/display/WW/FAQ
[7/4/2005 5:51 PM] <vitorsouz> That's it.
[7/4/2005 5:51 PM] <jaybose> ok, so first off, we need to order this better, and make it into
sections.
[7/4/2005 5:51 PM] <vitorsouz> Should we bother creating the FAQ sections now?
[7/4/2005 5:52 PM] <vitorsouz> Maybe we could think about it later.
```

```
[7/4/2005 5:52 PM] <jaybose> i agree, we'll tackle later
[7/4/2005 5:52 PM] <jaybose> I say skip Tutorial
[7/4/2005 5:53 PM] <jaybose> that should come based on Patrick's example app
[7/4/2005 5:53 PM] <vitorsouz> I haven't seen Patrick's example app.
[7/4/2005 5:53 PM] <jaybose> neither have I, that's why it should wait.
[7/4/2005 5:53 PM] <the_mind> i have only one comment
[7/4/2005 5:54 PM] <the_mind> i should not provide an example with scriptlets
[7/4/2005 5:54 PM] <the_mind> otherwise in terms of evolution of the tutorial it seems oke to
me
[7/4/2005 5:54 PM] <jaybose> scriplets?
[7/4/2005 5:54 PM] <vitorsouz> Ok. I'll talk to Patrick, then.
[7/4/2005 5:54 PM] <the_mind> Understanding actions (includes note on displaying data using
Scriptlets)
[7/4/2005 5:54 PM] <jaybose> ok
[7/4/2005 5:55 PM] <the_mind> let's wait the example app and than we will fix this one
[7/4/2005 5:55 PM] <jaybose> Moving to Cookbook: what type of docs should go in here?
[7/4/2005 5:55 PM] <the_mind> is Patrick working on it?
[7/4/2005 5:55 PM] <the_mind> or should we?
[7/4/2005 5:55 PM] <jaybose> he is, or is done.
[7/4/2005 5:55 PM] <the_mind> i have noticed something in cvs but not sure yet
[7/4/2005 5:56 PM] <vitorsouz> I'll talk to him. If he likes any of my suggestions on the
tutorial, I'll help him develop the example app to conform with the tutorial.
[7/4/2005 5:56 PM] <vitorsouz> And vice-versa.
[7/4/2005 5:56 PM] <jaybose> Ok. So far you have the following for Cookbook:
Tips and tricks on Application Servers (this was in "Overview")
Accessing application, session and request objects;
How to format dates and numbers;
Other stuff from the revised Cookbook...
[7/4/2005 5:56 PM] <the_mind> yes there is something in cvs
[7/4/2005 5:57 PM] <the_mind> see webwork-example dir
[7/4/2005 5:57 PM] <vitorsouz> Yeah, my suggestions on the Cookbook were detailed in the
paragraph and table that follows the bulleted list.
[7/4/2005 5:57 PM] <vitorsouz> I think the lessons in the current cookbook should be revised.
[7/4/2005 5:58 PM] <vitorsouz> Basic ones should be moved to tutorial.
[7/4/2005 5:58 PM] <the_mind> i would include in cookbook: alt syntax
[7/4/2005 5:58 PM] <jaybose> Ok, saw it.
[7/4/2005 5:58 PM] <the_mind> extensive validation
[7/4/2005 5:58 PM] <the_mind> custom interceptors
[7/4/2005 5:58 PM] <jaybose> he's a Q: what should go into the FAQ?
[7/4/2005 5:58 PM] <the_mind> we should extract teh faq from the ML
[7/4/2005 5:59 PM] <the_mind> browse a little the forum and identify the most asked questions
[7/4/2005 5:59 PM] <the_mind> that's what i would like to see in a faq
[7/4/2005 5:59 PM] <jaybose> I see the diff b/w cookbook and tutorial; but what's the diff b/w
these and the FAQ?
[7/4/2005 5:59 PM] <the_mind> shortcuts
[7/4/2005 5:59 PM] <jaybose> Maybe level of complexity?
[7/4/2005 5:59 PM] <vitorsouz> Maybe.
[7/4/2005 5:59 PM] <jaybose> shortcuts?
[7/4/2005 5:59 PM] <the_mind> the faq should be a shortcut to forum questions
[7/4/2005 5:59 PM] <the_mind> and to tricky parts
[7/4/2005 5:59 PM] <vitorsouz> A cookbook is more detailed than the FAQ.
[7/4/2005 6:00 PM] <the_mind> absolutely
[7/4/2005 6:00 PM] <the_mind> a cookbook is offering internals
[7/4/2005 6:00 PM] <the_mind> a faq: as the name says: frequently asked questions... this is
why i would look in the ML for the FAQ points
[7/4/2005 6:01 PM] <jaybose> Yeah, but we'd enter the Faq questions. They'd be entered after a
bunch of ppl as the same question.
[7/4/2005 6:02 PM] <jaybose> ok, I guess the Faq could also point to Cookbook answers
[7/4/2005 6:02 PM] <vitorsouz> I think what Jay is trying to say is: "How do I integrate WW
with Spring" is a FAQ. A lot of people ask it.
[7/4/2005 6:02 PM] <the_mind> yes
[7/4/2005 6:02 PM] <vitorsouz> Exactly: link from the FAQ to the cookbook.
[7/4/2005 6:02 PM] <the_mind> how do i pass parameters in xwork.xml
[7/4/2005 6:02 PM] <the_mind> and so on... this can be links to the cookbook or tutorial
[7/4/2005 6:03 PM] <the_mind> these*
[7/4/2005 6:03 PM] <vitorsouz> Alright.
[7/4/2005 6:03 PM] <jaybose> Related Projects: ...
[7/4/2005 6:03 PM] <vitorsouz> So, do we think about what the Cookbook should have now or just
leave the suggestion as it is?
[7/4/2005 6:03 PM] <jaybose> (skipping ref for a sec)
[7/4/2005 6:03 PM] <jaybose> the suggestion seems good enough
[7/4/2005 6:04 PM] <jaybose> it's clear what needs to be done
[7/4/2005 6:04 PM] <vitorsouz> Ok. Related Projects then.
[7/4/2005 6:04 PM] <jaybose> the person who takes that section will have to figure it out
[7/4/2005 6:04 PM] <the_mind> i think the info in cookbook should include almost everything in
the wiki right now that will not be covered in the tutorial
[7/4/2005 6:05 PM] <the_mind> it will be by far the toughest job
```

```
[7/4/2005 6:05 PM] <vitorsouz> Certainly.
[7/4/2005 6:05 PM] <the_mind> i don't think one single guy can do it
[7/4/2005 6:05 PM] <vitorsouz> Related Projects include information on other projects, like
WebFlow, Plugins, Optional modules, etc.
[7/4/2005 6:05 PM] <jaybose> Vitor has:
WebFlow (graphical chart tool)
EclipseWork (Eclipse Plugin)
IDEA Plugin
WebWork Optional
[7/4/2005 6:05 PM] <the_mind> i think at least 2 should work on it
[7/4/2005 6:05 PM] <the_mind> even in parallel
[7/4/2005 6:05 PM] <jaybose> the_mind: that will probably happen.
[7/4/2005 6:05 PM] <the_mind> why are you orange?
[7/4/2005 6:06 PM] <the_mind> :-/
[7/4/2005 6:06 PM] <vitorsouz> So, anything to add or change in the "Related Projects"?
[7/4/2005 6:06 PM] <jaybose> No, these seem pretty self-explanatory
[7/4/2005 6:07 PM] <vitorsouz> Ok. Can we start the discussion about the Reference, then?
[7/4/2005 6:07 PM] <jaybose> yep.
[7/4/2005 6:07 PM] <jaybose> this is what i had:
[7/4/2005 6:07 PM] <vitorsouz> Ok. There's your suggestion and there's my comments on it.
[7/4/2005 6:07 PM] <jaybose> What is Webwork - also explain what Webwork is not.
Architecture
Getting Started/Deployment Notes
Configuration
Interceptors
Action Chaining
IOC
JSP & Velocity Tags - this would be the current tag information, combined with the current "JSP
Expression Language Comparison with Webwork 1.x" pages
Webwork Freemarker Support
Result Types
Type Conversion
Validation
OGNL
Internationalization
Webflow
3rd Party Integration
[7/4/2005 6:07 PM] <jaybose> I like Jay Bose's reference TOC (below). On top of it, I'd
suggest:

 Remove "What is WebWork" and "Getting Started/Deployment Notes";

 Replace them with "Introduction", in which there would be instructions to read the Overview
first and proceed to the tutorial if the reader wants to get started;

 Append "/ Dependency Injection" to "IoC", because some people may know it only by the name
"Dependency Injection";

 Group "JSP & Velocity Tags" with "WebWork Freemarker Support" to create a single section that
contains everything related to user interface (in subsections);

 "JSP Expression Language Comparison with WebWork 1.x" should be in "Migrating from WebWork
1.x", in the "Project Information" section;

 Add to that same UI topic: JavaScript validation and DWR support (is this in 2.2?);

 WebFlow would be in "Related Projects";

 Have "3rd Party Integration" be links to Cookbook pages that explain how to integrate with
SiteMesh, Spring, Pico, Hibernate, JUnit, Quartz, etc.


[7/4/2005 6:07 PM] <the_mind> i must go to sleep now... sorry 2am
[7/4/2005 6:08 PM] <jaybose> goodnight
[7/4/2005 6:08 PM] <vitorsouz> Ok, the_mind: good night. Thanks for your opinions.
[7/4/2005 6:08 PM] <the_mind> jay pls send me the log
[7/4/2005 6:08 PM] <jaybose> it will be on the forum, i'll send an email
[7/4/2005 6:08 PM] <the_mind> i will comment on it... but as far as can say you are moving
pretty well without me
[7/4/2005 6:08 PM] <the_mind> :)
[7/4/2005 6:08 PM] <the_mind> good nite
[7/4/2005 6:08 PM] <vitorsouz> Night
[7/4/2005 6:09 PM] <jaybose> Vitor: give me a minute, i'll give a hybrid of your comments on it
[7/4/2005 6:09 PM] |<-- the_mind has left efnet (http://chat.efnet.org)
[7/4/2005 6:09 PM] <vitorsouz> Ok. I will hold.
[7/4/2005 6:10 PM] <vitorsouz> But since I'm not a native english speaker, I don't know what
you meant by "give a hybrid of my comments" :)
```

```
[7/4/2005 6:14 PM] <jaybose> i am made changes to my suggestions, based on your comments
[7/4/2005 6:15 PM] <jaybose> Introduction - which will have parts of Overview in it, rather
than forwarding them to Overview altogether.
Architecture
Configuration
Interceptors
Action Chaining
IOC / Dependency Injection
UI Components - JSP, Velocity, Freemarker, JavaScript validation and DWR support
Result Types
Type Conversion
Validation
OGNL
Internationalization
3rd Party Integration - SiteMesh, Spring, Pico, Hibernate, JUnit, Quartz, etc.
[7/4/2005 6:15 PM] <vitorsouz> Did you update the wiki page?
[7/4/2005 6:15 PM] <jaybose> now, i did not really chg the 3rd party, and i am trying to add
parts of Overview to the Intro
[7/4/2005 6:15 PM] <jaybose> not yet
[7/4/2005 6:15 PM] <vitorsouz> Ok, let me take a look at it here then.
[7/4/2005 6:16 PM] <jaybose> the reason for this is i believe the ref should be a doc that ppl
could print on it's own and pass around the office
[7/4/2005 6:16 PM] <jaybose> wiki is great, but not as simple to handle as a complete product
doc in PDF manual
[7/4/2005 6:16 PM] <jaybose> similar to Spring and Hibernate
[7/4/2005 6:17 PM] <jaybose> i think WW should move in that direction
[7/4/2005 6:17 PM] <vitorsouz> Ok. Agreed.
[7/4/2005 6:17 PM] <vitorsouz> I think the merge is good.
[7/4/2005 6:18 PM] <vitorsouz> About 3rd party info: it would not be links to the cookbook,
then?
[7/4/2005 6:18 PM] <jaybose> So there could be links to things within the ref, but it should a
standalone document. If a user needs more info on a subject, then they should look to the
tutorial or cookbook.
[7/4/2005 6:18 PM] <vitorsouz> Ok.
[7/4/2005 6:18 PM] <jaybose> no, b/c that is something ppl use a lot
[7/4/2005 6:19 PM] <vitorsouz> Should we define, then, that the Reference doesn't link to
anyone, people link to the reference?
[7/4/2005 6:19 PM] <jaybose> i know when i was adding Spring to my app, i looked at it's
Hibnernate support section constantly
[7/4/2005 6:19 PM] <jaybose> hmm, maybe
[7/4/2005 6:19 PM] <vitorsouz> So if "Spring integration in WebWork" is a FAQ and a Cookbook,
they all link to the reference, which will be written book-style.
[7/4/2005 6:19 PM] <jaybose> i think the reference could tell ppl where to find more
information: cookbook, tutorial
[7/4/2005 6:19 PM] <vitorsouz> If not book-style, hibernate-reference-style.
[7/4/2005 6:19 PM] <jaybose> yes.
[7/4/2005 6:20 PM] <vitorsouz> But would Confluence convert links to other pages to full URLs?
[7/4/2005 6:20 PM] <vitorsouz> When generating PDF?
[7/4/2005 6:21 PM] <vitorsouz> Can we ask Confluence to generate only the Reference as PDF and
all of its links to outside be converted?
[7/4/2005 6:21 PM] <jaybose> Why would not want the links to stay as is?
[7/4/2005 6:21 PM] <jaybose> Should the links change?
[7/4/2005 6:21 PM] <vitorsouz> Because if the Cookbook is not generated as PDF to be printed
and passed around, we need to change the links to the cookbook to their complete URL.
[7/4/2005 6:22 PM] <vitorsouz> Or am I missing something?
[7/4/2005 6:22 PM] <jaybose> ahhhh
[7/4/2005 6:22 PM] <jaybose> now i understand
[7/4/2005 6:22 PM] <jaybose> so the links are relative now?
[7/4/2005 6:23 PM] <jaybose> or you mean, you only see the name w/ an underline
[7/4/2005 6:23 PM] <vitorsouz> Yes.
[7/4/2005 6:23 PM] <vitorsouz> But not only that.
[7/4/2005 6:23 PM] <vitorsouz> When you link from one page to another in Confluence you just
have to write the page's name.
[7/4/2005 6:23 PM] <vitorsouz> For example, the TOC Homework page references my suggestions on
syle.
[7/4/2005 6:24 PM] <vitorsouz> Like this: [Click here|Style Guide]
[7/4/2005 6:24 PM] <vitorsouz> Click here is the text that will be underlined.
[7/4/2005 6:24 PM] <vitorsouz> The rest is the name of the page, which serves as URL.
[7/4/2005 6:24 PM] <jaybose> We can't control that, most references do that. If they need to
see what a link would lead to, they have to go back to the pdf and click on the link.
[7/4/2005 6:24 PM] <jaybose> we need to ask patrick if we can make the links full url's under
the name
[7/4/2005 6:24 PM] <jaybose> i bet not. crap.
[7/4/2005 6:25 PM] <vitorsouz> Ok. Also we have to ask him how do we organize the sections in a
way that we can choose to generate PDF only for the Reference, not the other pages.
[7/4/2005 6:25 PM] <jaybose> By the way, i hate confluence.
[7/4/2005 6:25 PM] <jaybose> ok, so we'll need to see what he has to say about this.
```

```
[7/4/2005 6:25 PM] <vitorsouz> Are you going to cross that phrase from the log before
publsihing it? haha :)
[7/4/2005 6:26 PM] <jaybose> haha
[7/4/2005 6:26 PM] <jaybose> maybe i should... :-J
[7/4/2005 6:26 PM] <vitorsouz> ;)
[7/4/2005 6:26 PM] <jaybose> ok, so any other ideas, if we can't convert links?
[7/4/2005 6:26 PM] <jaybose> just in case?
[7/4/2005 6:26 PM] <vitorsouz> Not reference anything.
[7/4/2005 6:27 PM] <vitorsouz> Invesion of Control... Everybody references the Ref, the Ref
references no one. :P
[7/4/2005 6:27 PM] <jaybose> hmm, ok. Maybe just reference sections by name.
[7/4/2005 6:27 PM] <jaybose> right.
[7/4/2005 6:27 PM] <vitorsouz> Or that.
[7/4/2005 6:27 PM] <jaybose> haha
[7/4/2005 6:27 PM] <jaybose> ok, i think we've made some progress
[7/4/2005 6:28 PM] <vitorsouz> Definetly.
[7/4/2005 6:28 PM] <jaybose> I'll the notes i made to the bottom of that Page you made
[7/4/2005 6:28 PM] <vitorsouz> Ok. Are you doing that now?
[7/4/2005 6:28 PM] <jaybose> and publish this conversation, and then send out a note
[7/4/2005 6:28 PM] <jaybose> yep
[7/4/2005 6:28 PM] <vitorsouz> We could change this "Someone's Suggestions" page to Doc Team
Suggestions.
[7/4/2005 6:29 PM] <jaybose> will do
[7/4/2005 6:29 PM] <vitorsouz> Move your Reference section instead of mine and add the other
notes.
[7/4/2005 6:29 PM] <jaybose> oh you want me to chg the entire page?
[7/4/2005 6:29 PM] <jaybose> i was going to append
[7/4/2005 6:29 PM] <vitorsouz> Yes.
[7/4/2005 6:30 PM] <vitorsouz> If you want to append I can reorganize it later.
[7/4/2005 6:30 PM] <vitorsouz> Just let me know when you're done.
[7/4/2005 6:30 PM] <vitorsouz> Confluence should have support for simultaneous work! :) haha
[7/4/2005 6:30 PM] <jaybose> :)
[7/4/2005 6:30 PM] <jaybose> agreed
[7/4/2005 6:36 PM] <jaybose> Done. Check it out and update as needed. Thanks for the patience.
[7/4/2005 6:37 PM] <vitorsouz> Ok, checking it now.
[7/4/2005 6:37 PM] <vitorsouz> This new IRC client that I got (VIRC) doesn't notify on channel
messages...
[7/4/2005 6:38 PM] <vitorsouz> Ok. I will reorganize the page, if you don't mind. Alright?
[7/4/2005 6:39 PM] <jaybose> Catch you on the forums; see ya.
[7/4/2005 6:39 PM] <jaybose> np.
```

Comparison to other web frameworks

This page last changed on Apr 25, 2006 by scud.

- 与Struts比较
- Comparison to JSF 译注:空内容
- Comparison to Tapestry 译注:空内容
- Comparison to Spring MVC 译注:空内容
- 与Ruby on Rails比较

Matt Raible 编写了 (mid 2005) 一个有趣的报告,比较了不同的框架. 你可以在这里查看PDF:
https://equinox.dev.java.net/framework-comparison/WebFrameworks.pdf
http://www.virtuas.com/files/osl-jwf-01.pdf

Comparison to JSF

This page last changed on Mar 30, 2006 by scud.

TODO: a brief write-up comparing WebWork to JSF

Comparison to Ruby on Rails

This page last changed on Mar 30, 2006 by scud.

WebWork's architecture is very similar to Ruby on Rails. The biggest difference between WebWork and Rails is actually more of a difference between Java and Ruby than anything. Using FreeMarker and QuickStart, developers can achieve the same level of productivity as developers who use Rails and the fact that Ruby is a scripting language.

Comparison to Spring MVC

This page last changed on Mar 30, 2006 by scud.

TODO: a brief write-up comparing WebWork to Spring MVC

Comparison to Struts

This page last changed on Apr 25, 2006 by scud.

⚠ 2005年12月,struts和webwork团队同意合并共同开发Struts Action 2.0, Struts Action 2.0 基于 WebWork 2.2. 所以,下面的比较与这两个社区的合作无关

性能比较

Feature	Struts	WebWork 1.x	WebWork 2.x
Action classes	Struts 要求Action类继承一个抽象类.这是Struts针对抽象类而不是接口编程所造成的一个很普遍的问题.	Action 类必须实现webwork.Action接口. 也可以实现提供其他服务的接口,如存储错误消息,取得本地化文本等. ActionSupport类实现了其中的一些接口,可以作为一个基类来使用. WebWork全部针对接口编程,这样很容易插入你自己的实现.	与WebWork 1.x相同,Action必须实现com.opensymphony.xwork.Action接口, 以及一系列的提供其他服务的接口. WebWork2提供了ActionSupport类实现这些接口.
Threading Model	Struts Actions必须是线程安全的,因为Action在处理所有的请求时都只有一个实例.这要求和Struts Actions一起处理的所有资源都要是线程安全的,或者是同步的.	WebWork Actions 为每个请求初始化一个实例, 这样就解决了线程安全的问题. 实际上, Servlet 容器每次处理请求时,都会生成许多将被丢弃的对象,但并没有证据证明更多的对象会产生性能或者垃圾回收问题	同WebWork 1.x
Servlet Dependency	Struts Actions 依赖于Servlets ,因为Struts Actions 在执行的时候使用了ServletRequest and ServletResponse (并不是HttpServletRequest and HttpServletResponse).这种对于Servlets的依赖事实上是对Servlets容器的依赖,这样的依赖是不必要的.例如, Servlet可以在Web环境之外运行, 但JMS就不能.	WebWork Actions 不依赖于web和容器 WebWork actions可以从ActionContext访问request和response对象,但是这不是必须的,并且最好在绝对需要的时候才去做,以避免对于web层的依赖.	同WebWork 1.x
Testability	已经有许多测试Struts应用的策略,其中最大的障碍是由于Struts Actions对于web层的紧密依赖.(使用Request和Response对象). 这导致人们需要在容器内测试Struts Actions.这样做即缓慢又称不上是单元测试. 有一个Junit的扩展 : Struts TestCase (http://strutstestcase.sourceforge.net/)	WebWork actions 通过初始化,设置属性,运行你的Action可以很容易的进行测试	同WebWork 1.x, 由于加强了控制翻转(Inversion of Control)的应用使得测试更加简单.你不需要注册服务或者使用静态单例(static singleton),仅仅向你的Action里注入实现服务接口的模拟对象(Mock)就可以进行测试
FormBeans	Struts需要对每个表单创建FormBeans, 使得要么增加许多额外的类,要么使用DynaBeans, 而DynaBeans恰恰是这一限制的产物	WebWork 1.x 允许你像取得一般的Javabeans的属性那样直接取得你Action中的所有属性, 属性可以包含丰富的对象类型甚至可以有自己的属性, 而这些属性都可以从Web页面中访问WebWork也	WebWork 2 拥有WebWork 1相同的特性, 除此之外,还添加了模型驱动(ModelDriven)Action.模型驱动Action允许你把富对象类型或者是域对象当做form bean使用,这样它的属性可

		支持FormBean模式，在"Populate Form Bean and access its value"中,有相关的论述	以在web页面上直接访问,而不用做为Action的属性的一个子属性来访问.
Expression Language	Struts 1.1 集成 JSTL，因此可以在Struts中使用JSTL EL. 这种EL语言只能进行基本的对象图的遍历,对于集合和索引属性方面的支持要弱一些.	WebWork 1.x拥有自己的表达式语言用以访问值堆栈. 对集合以及索引属性的提供很基本的支持但工作的很好.WebWork也能和JSTL一起工作	WebWork 使用Ognl,一个非常强大的表达式语言，可以访问值堆栈. Ognl 对集合和索引属性提供强有力的支持. Ognl 还拥有其他强大的特性,例如projections (调用集合中所有成员的相同方法,把结果构造成一个新的集合), selections (用一个选择表达式过滤一个集合并返回得到的子集), list construction, and lambda expressions (可以重用的简单函数). Ognl 也可以访问静态方法,静态字段以及类的构造器. WebWork2也能使用JSTL，参见Using JSTL seamlessly with WebWork
Binding values into views	Struts采用标准的JSP机制来把对象绑定到页面(page context),这使得视图与formBean紧密耦合	WebWork 使用可以被WebWork的标签库直接访问的值堆栈，显得更加灵活并且不会把视图和你要生成的类型绑定起来.这样就可以在很大的范围内复用视图.	同WebWork 1.x
Type Conversion	Struts FormBeans 的属性通常都是String类型的. Struts使用Commons-Beanutils进行类型转换.每个类使用一个转换器，但不允许为每个实例配置不同的转换器. 所以很难得到一个有意义的类型转换错误并把它显示给用户.	WebWork 1.x应用PropertyEditors来进行类型转换. PropertyEditors针对类型而不是Action，但是字段的错误信息会被加进Action中的字段错误表(field error map)中,并且自动的显示给用户.	WebWork2 应用Ognl 进行类型转换,Ognl提供针对基本类型的转换器. 类型转换缺省情况下使用上述转换器，但也可以为每个类的每个字段指定转换器. 类型转换有默认的错误消息,但是也可以应用WW2的本地化机制来为每个类的每个字段设置一个错误消息.这些消息会被装载到Action的字段错误消息中.
Modular Before & After Processing	必须创建一个继承基类Action的类来作为Action前/后处理的代理，这会导致类继承层次过深和对多重继承的限制 Comparison to Struts#1	类层次	WebWork 2 允许你在拦截器里面模块化这种处理.拦截器可以通过配置动态的使用而无需和Action类产生耦合.
Validation	Struts调用FormBean中的validate()方法,Struts的使用者一般使用Commons Validation来进行验证.我对此不太了解，因此仅提出几个问题: 由于FormBean的属性一般是String类型的,某些校验方法是否会重复或无法做到? Commons Validation是否能为同一个类设置不同的校验环境?(我已被告知可以做到，这是好事) Commons Validation能否使	WebWork1.x 调用Action中的validate()方法，即可以手工编写验证也可以调用外部的验证框架(这显然和Struts相同)	WebWork2可以使用类似与WebWork和Struts那样使用validate()方法来进行验证,也可以使用这样一个由XWork拦截器激活的验证框架. Xwork验证框架允许用XML格式编写校验器,根据不同的上下文增加自定义验证. Xwork验证框架通过拦截器生效因此和Action完全解耦. Xwork验证框架也允许你使用VisitorFieldValidator对子属性进行持续验证

	用为对象属性类定义的校验方式对子对象(sub-objects)进行校验?		,VisitorFieldValidato利用了为类的属性和上下文定义的验证信息.
Control Of Action Execution	就我所知,Struts为你设置好了Action对象,因此你很难去控制执行的顺序. 为了改变执行的顺序我认为?你需要写一个自己的Servlet来处理你需要的分派规则	Action工厂(ActionFactory)控制了Action构建和初始化的顺序,但需要编写一个类	在这点上WebWork 2的拦截器堆栈显得特别强大. Action设置的方方面面都被移到了拦截器的实现中(例如从web设置参数,验证等等), 所以你能根据它们执行顺序控制每一个Action. 例如你想在参数从request设置之前用你的IOC框架来配置action或者进行相反的操作, - 也可以使用截取器栈对每个包或每个类进行控制.

参考

- http://www.mail-archive.com/opensymphony-webwork@lists.sourceforge.net/msg00995.html - 一个从Struts转向WebWork的开发者对于Struts开发和WebWork开发的比较
- http://www.mail-archive.com/opensymphony-webwork@lists.sourceforge.net/msg04700.html - 本文的草稿

脚注

1. 一些Struts的用户为Struts开发了一套拦截器框架 (http://struts.sourceforge.net/saif/). 但目前还有一些很严重的局限 (不能进行"包围(around)"处理, 子允许前/后处理) 而且也不是Struts项目的一部分

Comparison to Tapestry

This page last changed on Mar 30, 2006 by scud.

TODO: a brief write-up comparing WebWork to Tapestry

Configuration Files

This page last changed on May 19, 2006 by scud.

主要配置文件

WebWork有两个主要配置文件:web.xml 和 xwork.xml.在下面您可以找到WebWork必须和可选的配置文件的所有信息.

下面是您需要注意的所有文件.为了使开发更容易,其中的一些是可以动态重新加载的.更多信息请查看重载配置.

文件	可选	位置(相对于webapp)	用途
web.xml	否	/WEB-INF/	Web部署描述,包括所有必须的WebWork组件
xwork.xml	否	/WEB-INF/classes/	主要配置,包括result/view类型,action影射,拦截器等
webwork.properties	是	/WEB-INF/classes/	WebWork的属性配置
webwork-default.xml	是	/WEB-INF/lib/webwork-x.x.jar	xwork.xml中应该有的默认配置
velocity.properties	是	/WEB-INF/classes/	velocity配置

静态内容

webwork需要的公共静态内容(JavaScript和CSS文件等)是由FilterDispatcher过滤器自动提供的.任何以"/webwork/"开始的请求被认为是静态内容,"/webwork/"后面的值会被影射到classpath下的WebWork公共包

默认情况下,系统会搜索以下包:

- com.opensymphony.webwork.static
- template

附加包可以通过设定packages参数(web.xml中FilterDispatcher过滤器的配置)来指定,这个参数是以逗号分割的列表.当指定附加的静态内容时,注意不要暴露敏感信息(如,数据库密码等).

Reloading configuration

This page last changed on May 18, 2006 by scud.

WebWork允许动态装载xml配置文件(例如,重新装入actions.xml).

这允许你在开发过程中重新配置你的action映射.也许这会有一点轻微的性能损失,因为不推荐你在生产阶段使用.

为了开启这个特性,在你的webwork.properties文件里添加下面的内容:

```
webwork.configuration.xml.reload=true
```

web.xml

This page last changed on May 18, 2006 by scud.

如果要用到WebWork的最新特性并且不需要考虑向后兼容问题,您只需要在web.xml中添加一个单独的过滤器.如果要用JSP的话,还需要一个添加标签库.但是如果是从2.1.7或更早的版本升级的Web应用程序,那么需要做更多的工作使程序能够正常运行.更多信息请查看web.xml 2.1.x兼容性.

过滤器配置如下:

```
<filter>
    <filter-name>webwork</filter-name>
    <filter-class>com.opensymphony.webwork.dispatcher.FilterDispatcher</filter-class>
</filter>
<filter-mapping>
    <filter-name>webwork</filter-name>
    <url-pattern>/*</url-pattern>
</filter-mapping>
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
```

如果要用到JSP,标签库配置如下:

```
<!--
    #############,###########webwork.jar##.
    ########web.xml######,###webwork/src/java/META-INF/taglib.tld###########WEB-INF###,###
webwork.tld
 -->
<taglib>
    <taglib-uri>webwork</taglib-uri>
    <taglib-location>/WEB-INF/webwork.tld</taglib-location>
</taglib>
```

译者注:如果是servlet2.4,标签库配置如下:

```
<jsp-config>
   <taglib>
      <taglib-uri>webwork</taglib-uri>
      <taglib-location>/WEB-INF/webwork.tld</taglib-location>
   </taglib>
</jsp-config>
```

⚠ 如果你要使用 SiteMesh 进行页面装饰,你应该需要添加一些 额外的过滤器

web.xml 2.1.x compatibility

This page last changed on May 18, 2006 by scud.

在WebWork 2.2之前,一个ServletDispatcher 用来处理action请求.另外,在Velocity里模拟了JSP标签.WebWork在这里做了一个重要的改变:ServletDispatcher 不被推荐使用,被FilterDispatcher代替.这对用那些遵循WebWork最佳实践的用户来说通常工作的很好,也是2.2版本推荐的.然而,因为WebWork 2.2中一些小的行为的变化,老的应用程序可能需要ServletDispatcher.

最大的需要注意的变化是任何包含了其他action的应用,不管是通过一个result dispatcher还是jsp/ww:include标签,使用FilterDispatcher时将不再工作.这是因为Servlet容器不支持请求分发器转向到filter的映射 - 仅有servlet的映射被支持. 为了避开这个问题,你可以改变你的代码来使用action链来代替result分发,以及使用ww:action标签来替换jsp/ww:include.

译注:在Servlet 2.4中,通过设置可以使用原来的方法,尽管这是不推荐的.详细可阅读 http://www.jscud.com/srun/news/viewhtml/4_2006_4/188.htm .

作为一个切换到FilterDispatcher的结果,在Velocity里面的JSP标签仿真不能工作了.这个特性也从来没有非常好地工作和被支持,我们承认(recognize)需要用户利用了这个特性.在WebWork 2.2中,原生的Velocity标签被提供了,而且是WebWork/Velocity 集成方式中唯一支持的标签.

当然,我们也提供了一个不推荐的方式来避免改变你的代码.我们推荐只要有可能你就更新你的代码.同时,你可能添加下面的Servlets到 web.xml:

```
<servlet>
    <servlet-name>JspSupportServlet</servlet-name>
    <servlet-class>com.opensymphony.webwork.views.JspSupportServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>

<servlet>
    <servlet-name>action</servlet-name>
    <servlet-class>com.opensymphony.webwork.dispatcher.ServletDispatcher</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>action</servlet-name>
    <url-pattern>*.action</url-pattern>
</servlet-mapping>
```

webwork-default.xml

This page last changed on May 18, 2006 by scud.

在webwork jar文件中有一个名为webwork-default.xml的基础配置文件.这个文件可以在xwork.xml文件的顶部引用,这样就加载标准配置而不需要复制其内容,像这样:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd"><xwork>
    <include file="webwork-default.xml"/>

    <package name="default" extends="webwork-default">
    ...
    </package>
</xwork>
```

webwork-default.xml 的内容如下:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.1.1//EN"
"http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">
<xwork>
    <package name="webwork-default">
        <result-types>
            <result-type name="chain" class="com.opensymphony.xwork.ActionChainResult"/>
            <result-type name="dispatcher"
class="com.opensymphony.webwork.dispatcher.ServletDispatcherResult" default="true"/>
            <result-type name="freemarker"
class="com.opensymphony.webwork.views.freemarker.FreemarkerResult"/>
            <result-type name="httpheader"
class="com.opensymphony.webwork.dispatcher.HttpHeaderResult"/>
            <result-type name="jasper"
class="com.opensymphony.webwork.views.jasperreports.JasperReportsResult"/>
            <result-type name="redirect"
class="com.opensymphony.webwork.dispatcher.ServletRedirectResult"/>
            <result-type name="redirect-action"
class="com.opensymphony.webwork.dispatcher.ServletActionRedirectResult"/>
            <result-type name="stream"
class="com.opensymphony.webwork.dispatcher.StreamResult"/>
            <result-type name="tiles"
class="com.opensymphony.webwork.views.tiles.TilesResult"/>
            <result-type name="velocity"
class="com.opensymphony.webwork.dispatcher.VelocityResult"/>
            <result-type name="xslt" class="com.opensymphony.webwork.views.xslt.XSLTResult"/>
            <result-type name="plaintext"
class="com.opensymphony.webwork.dispatcher.PlainTextResult" />

            <!-- Results necessary when using 'browse server' and 'upload' feature of
Richtexteditor -->
            <result-type name="richtexteditorGetFolders"
class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorGetFoldersResult" />
            <result-type name="richtexteditorGetFoldersAndFiles"
class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorGetFoldersAndFilesResult" />
            <result-type name="richtexteditorCreateFolder"
class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorCreateFolderResult" />
            <result-type name="richtexteditorFileUpload"
class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorFileUploadResult" />

        </result-types>

        <interceptors>
            <interceptor name="alias"
class="com.opensymphony.xwork.interceptor.AliasInterceptor"/>
            <interceptor name="autowiring"
class="com.opensymphony.xwork.spring.interceptor.ActionAutowiringInterceptor"/>
            <interceptor name="chain"
class="com.opensymphony.xwork.interceptor.ChainingInterceptor"/>
            <interceptor name="component"
class="com.opensymphony.xwork.interceptor.component.ComponentInterceptor"/>
            <interceptor name="conversionError"
class="com.opensymphony.webwork.interceptor.WebWorkConversionErrorInterceptor"/>
```

```
            <interceptor name="createSession"
class="com.opensymphony.webwork.interceptor.CreateSessionInterceptor" />
            <interceptor name="external-ref"
class="com.opensymphony.xwork.interceptor.ExternalReferencesInterceptor"/>
            <interceptor name="execAndWait"
class="com.opensymphony.webwork.interceptor.ExecuteAndWaitInterceptor"/>
            <interceptor name="exception"
class="com.opensymphony.xwork.interceptor.ExceptionMappingInterceptor"/>
            <interceptor name="fileUpload"
class="com.opensymphony.webwork.interceptor.FileUploadInterceptor"/>
            <interceptor name="i18n"
class="com.opensymphony.xwork.interceptor.I18nInterceptor"/>
            <interceptor name="logger"
class="com.opensymphony.xwork.interceptor.LoggingInterceptor"/>
            <interceptor name="model-driven"
class="com.opensymphony.xwork.interceptor.ModelDrivenInterceptor"/>
            <interceptor name="params"
class="com.opensymphony.xwork.interceptor.ParametersInterceptor"/>
            <interceptor name="prepare"
class="com.opensymphony.xwork.interceptor.PrepareInterceptor"/>
            <interceptor name="static-params"
class="com.opensymphony.xwork.interceptor.StaticParametersInterceptor"/>
            <interceptor name="scope"
class="com.opensymphony.webwork.interceptor.ScopeInterceptor"/>
            <interceptor name="servlet-config"
class="com.opensymphony.webwork.interceptor.ServletConfigInterceptor"/>
            <interceptor name="sessionAutowiring"
class="com.opensymphony.webwork.spring.interceptor.SessionContextAutowiringInterceptor"/>
            <interceptor name="timer"
class="com.opensymphony.xwork.interceptor.TimerInterceptor"/>
            <interceptor name="token"
class="com.opensymphony.webwork.interceptor.TokenInterceptor"/>
            <interceptor name="token-session"
class="com.opensymphony.webwork.interceptor.TokenSessionStoreInterceptor"/>
            <interceptor name="validation"
class="com.opensymphony.xwork.validator.ValidationInterceptor"/>
            <interceptor name="workflow"
class="com.opensymphony.xwork.interceptor.DefaultWorkflowInterceptor"/>

            <!-- Basic stack -->
            <interceptor-stack name="basicStack">
                <interceptor-ref name="exception"/>
                <interceptor-ref name="servlet-config"/>
                <interceptor-ref name="prepare"/>
                <interceptor-ref name="static-params"/>
                <interceptor-ref name="params"/>
                <interceptor-ref name="conversionError"/>
            </interceptor-stack>

            <!-- Sample validation and workflow stack -->
            <interceptor-stack name="validationWorkflowStack">
                <interceptor-ref name="basicStack"/>
                <interceptor-ref name="validation"/>
                <interceptor-ref name="workflow"/>
            </interceptor-stack>

            <!-- Sample file upload stack -->
            <interceptor-stack name="fileUploadStack">
                <interceptor-ref name="fileUpload"/>
                <interceptor-ref name="basicStack"/>
            </interceptor-stack>

            <!-- Sample WebWork Inversion of Control stack
                 Note: WebWork's IoC is deprecated - please
                 look at alternatives such as Spring -->
            <interceptor-stack name="componentStack">
                <interceptor-ref name="component"/>
                <interceptor-ref name="basicStack"/>
            </interceptor-stack>

            <!-- Sample model-driven stack  -->
            <interceptor-stack name="modelDrivenStack">
                <interceptor-ref name="model-driven"/>
                <interceptor-ref name="basicStack"/>
            </interceptor-stack>

            <!-- Sample action chaining stack -->
            <interceptor-stack name="chainStack">
```

```
                <interceptor-ref name="chain"/>
                <interceptor-ref name="basicStack"/>
            </interceptor-stack>

            <!-- Sample i18n stack -->
            <interceptor-stack name="i18nStack">
                <interceptor-ref name="i18n"/>
                <interceptor-ref name="basicStack"/>
            </interceptor-stack>

            <!-- Sample execute and wait stack.
                 Note: execAndWait should always be the *last* interceptor. -->
            <interceptor-stack name="executeAndWaitStack">
                <interceptor-ref name="basicStack"/>
                <interceptor-ref name="execAndWait"/>
            </interceptor-stack>

            <!-- An example of the params-prepare-params trick. This stack
                 is exactly the same as the defaultStack, except that it
                 includes one extra interceptor before the prepare interceptor:
                 the params interceptor.

                 This is useful for when you wish to apply parameters directly
                 to an object that you wish to load externally (such as a DAO
                 or database or service layer), but can't load that object
                 until at least the ID parameter has been loaded. By loading
                 the parameters twice, you can retrieve the object in the
                 prepare() method, allowing the second params interceptor to
                 apply the values on the object. -->
            <interceptor-stack name="paramsPrepareParamsStack">
                <interceptor-ref name="exception"/>
                <interceptor-ref name="alias"/>
                <interceptor-ref name="params"/>
                <interceptor-ref name="servlet-config"/>
                <interceptor-ref name="prepare"/>
                <interceptor-ref name="i18n"/>
                <interceptor-ref name="chain"/>
                <interceptor-ref name="model-driven"/>
                <interceptor-ref name="fileUpload"/>
                <interceptor-ref name="static-params"/>
                <interceptor-ref name="params"/>
                <interceptor-ref name="conversionError"/>
                <interceptor-ref name="validation">
                    <param name="excludeMethods">input,back,cancel</param>
                </interceptor-ref>
                <interceptor-ref name="workflow">
                    <param name="excludeMethods">input,back,cancel</param>
                </interceptor-ref>
            </interceptor-stack>

            <!-- A complete stack with all the common interceptors in place.
                 Generally, this stack should be the one you use, though it
                 may do more than you need. Also, the ordering can be
                 switched around (ex: if you wish to have your servlet-related
                 objects applied before prepare() is called, you'd need to move
                 servlet-config interceptor up.

                 This stack also excludes from the normal validation and workflow
                 the method names input, back, and cancel. These typically are
                 associated with requests that should not be validated.
                 -->
            <interceptor-stack name="defaultStack">
                <interceptor-ref name="exception"/>
                <interceptor-ref name="alias"/>
                <interceptor-ref name="servlet-config"/>
                <interceptor-ref name="prepare"/>
                <interceptor-ref name="i18n"/>
                <interceptor-ref name="chain"/>
                <interceptor-ref name="model-driven"/>
                <interceptor-ref name="fileUpload"/>
                <interceptor-ref name="static-params"/>
                <interceptor-ref name="params"/>
                <interceptor-ref name="conversionError"/>
                <interceptor-ref name="validation">
                    <param name="excludeMethods">input,back,cancel,browse</param>
                </interceptor-ref>
                <interceptor-ref name="workflow">
                    <param name="excludeMethods">input,back,cancel,browse</param>
                </interceptor-ref>
            </interceptor-stack>

            <!-- The completeStack is here for backwards compatibility for
                 applications that still refer to the defaultStack by the
                 old name -->
            <interceptor-stack name="completeStack">
                <interceptor-ref name="defaultStack"/>
            </interceptor-stack>
        </interceptors>

        <default-interceptor-ref name="defaultStack"/>
    </package>
</xwork>
```

这个文件定义了所有内置的结果和拦截器.还定义了很多拦截器栈,这些栈可以原封不动的使用,也可以作为您自己的栈的基础. 注意包名是 "webwork-default".

webwork.properties

This page last changed on May 18, 2006 by scud.

WebWork有很多属性可以根据需要改变.要改变它们,请指定classpath(通常是/WEB-INF/classes)下的webwork.properties文件中的值.属性列表可以在default.properties文件中找到(在webwork.jar中):

译注:为了方便,翻译时把原文件拆开翻译,并对格式进行了一些改变.请自己打开default.properties了解原来的内容和格式.
default.properties的每一行如果前面有"#"符号,则表示该行为备注,有一些行仅仅是为了提示你如何使用,如果你需要使用,则要去掉相应的"#".

文件说明

default.properties是Webwork的缺省配置文件,可以被classpath根目录下的 webwork.properties 文件覆盖

配置工厂

指定用来配置webwork的 Configuration.
用户可以扩展 com.opensymphony.webwork.config.Configuration 来创建自己的方式来获取配置参数以传给webwork

```
# webwork.configuration=com.opensymphony.webwork.config.DefaultConfiguration
```

指定locale,编码

用来设置你的缺省 locale和编码方案

```
# webwork.locale=en_US
webwork.i18n.encoding=UTF-8
```

Object Factory

如果指定了,缺省的ojbect factory在这里可以被覆盖
注意:缩写在某些情况下是支持的,例如 "spring" .作为代替,你可以提供一个 com.opensymphony.xwork.ObjectFactory 子类的名称

```
# webwork.objectFactory = spring
```

自动装配策略

指定当使用SpringObjectFactory时的自动装配逻辑.
合法值包括: name, type, auto, 和 constructor (缺省为 name)

```
webwork.objectFactory.spring.autoWire = name
```

类缓存

标识webwork-spring集成,如果类实例应该被缓存
这可能,直到将来Spring版本让它变得可能,否则就保持它为true
除非你确切的知道你在做什么,否则不要改变
合法的值包括: true, false (true 是缺省的)

```
webwork.objectFactory.spring.useClassCache = true
```

缺省对象类型裁决者

如果设定了,缺省的对象类型裁决者可以被覆盖

注意: 缩写在某些情况下支持,例如 "tiger" or "notiger"
作为代替,你可以提供一个 com.opensymphony.xwork.util.ObjectTypeDeterminer 实现类的名字
注意: 如果在classpath里有 xwork-tiger.jar, GenericsObjectTypeDeterminer 缺省会被使用
关闭tiger支持,在这里使用 "notiger" 属性值.

```
#webwork.objectTypeDeterminer = tiger
#webwork.objectTypeDeterminer = notiger
```

文件上传设置

用来处理HTTP POST请求,编码使用MIME-type multipart/form-data方式的

```
# webwork.multipart.parser=cos
# webwork.multipart.parser=pell
webwork.multipart.parser=jakarta
```

1. 保存的目录缺省使用 javax.servlet.context.tempdir

```
webwork.multipart.saveDir=
webwork.multipart.maxSize=2097152
```

定制配置

装载定制的属性文件(不会覆盖webwork.properties!)

```
# webwork.custom.properties=application,com/webwork/extension/custom
```

缺省URL映射处理器

用来处理request URL如何映射到action,或者相反的映射(用户可以实现自己的处理)

```
webwork.mapper.class=com.opensymphony.webwork.dispatcher.mapper.DefaultActionMapper
```

缺省后缀

DefaultActionMapper使用的设置
你可以提供一个逗号分割的列表,例如 webwork.action.extension=action,jnlp,do

```
webwork.action.extension=action
```

是否静态文件由FilterDispatcher服务

FilterDispatcher使用的设置
如果为true,那么WebWork将会为它jar包内的静态文件提供服务.
如果为false,那么静态文件必须在<context_path>/webwork下面可以访问

```
webwork.serve.static=true
```

标签语法设定

使用 alternative syntax 在大多数地方需要 %{} 来计算标签的字符串属性的值

```
webwork.tag.altSyntax=true
```

开发模式设定

当设置为true时,WebWork会对开发者更友好.这包括:

- webwork.i18n.reload = true
- webwork.configuration.xml.reload = true
- 引发不同的调试信息或者忽略的问题为错误信息
  例如: 正常情况下,一个请求 foo.action?someUnknownField=true 会被忽略(从web过来的任何值,都是不可信的).
  因此,在开发的时候,当这些错误发生时立刻提醒,这可能是有用的

```
webwork.devMode = false
```

⚠ 译注:设置为true时,所有有名字(name)的input都必须有对应的setter,否则会报错.如果不需要,请删除name.

国际化资源设定

当设置为true,资源包会在每个请求时自动重新载入.
这在开发时很方便,但是不应该在生产状态下使用

```
webwork.i18n.reload=false
```

theme和模板设定

标准的UI theme
改变这个会映射缺省由那个路径的模板来输出JSP控件标签

```
webwork.ui.theme=xhtml
webwork.ui.templateDir=template
```

设置缺省模板类型,可以是ftl,vm,jsp

```
webwork.ui.templateSuffix=ftl
```

配置自动更新设定

配置自动重新装载,这会导致配置来重新载入xwork.xml, 当它改变的时候

```
webwork.configuration.xml.reload=false
```

Velocity配置设定

velocity.properties 文件的位置.缺省是 velocity.properties

```
# webwork.velocity.configfile = velocity.properties
```

逗号分割的 VelocityContext 类名来链接到 WebWorkVelocityContext

```
# webwork.velocity.contexts =
```

URL相关设定

用来构建URL,例如UrlTag

```
webwork.url.http.port = 80
webwork.url.https.port = 443
```

自定义国际化资源

装载自定义的缺省资源包(如果有多个,用逗号分割)

```
# webwork.custom.i18n.resources=testmessages,testmessages2
```

应用服务器相关设定

有些app server不能处理HttpServletRequest.getParameterMap(),经常使用的是WebLogic,Orion和OC4J

```
webwork.dispatcher.parametersWorkaround = false
```

FreeMarker相关设定

配置要用的 Freemarker Manager 类
允许用户插入自定义的 Freemarker Manager ,如果需要的话
必须扩展 com.opensymphony.webwork.views.freemarker.FreemarkerManager

```
#webwork.freemarker.manager.classname=com.opensymphony.webwork.views.freemarker.FreemarkerManager
```

浏览 WebWorkBeanWrapper 的javadoc 了解更多信息

```
webwork.freemarker.wrapper.altMap=true
```

XSLTResult相关设定

配置 XSLTResult 类使用 stylesheet 缓存.
开发时设置为true,生产时设置为false.

```
webwork.xslt.nocache=false
```

xwork.xml

This page last changed on May 18, 2006 by scud.

xwork dtd的例子

```
<!--
   XWork configuration DTD.
   Use the following DOCTYPE

   <!DOCTYPE xwork PUBLIC
        "-//OpenSymphony Group//XWork 1.1.1//EN"
 "http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">
-->

<!ELEMENT xwork (package|include)*>

<!ELEMENT package (result-types?, interceptors?, default-interceptor-ref?, default-action-ref?,
global-results?, global-exception-mappings?, action*)>
<!ATTLIST package
    name CDATA #REQUIRED
    extends CDATA #IMPLIED
    namespace CDATA #IMPLIED
    abstract CDATA #IMPLIED
    externalReferenceResolver NMTOKEN #IMPLIED
>

<!ELEMENT result-types (result-type+)>

<!ELEMENT result-type (param*)>
<!ATTLIST result-type
    name CDATA #REQUIRED
    class CDATA #REQUIRED
    default (true|false) "false"
>

<!ELEMENT interceptors (interceptor|interceptor-stack)+>

<!ELEMENT interceptor (param*)>
<!ATTLIST interceptor
    name CDATA #REQUIRED
    class CDATA #REQUIRED
>

<!ELEMENT interceptor-stack (interceptor-ref+)>
<!ATTLIST interceptor-stack
    name CDATA #REQUIRED
>

<!ELEMENT interceptor-ref (param*)>
<!ATTLIST interceptor-ref
    name CDATA #REQUIRED
>

<!ELEMENT default-interceptor-ref (param*)>
<!ATTLIST default-interceptor-ref
    name CDATA #REQUIRED
>

<!ELEMENT default-action-ref (param*)>
<!ATTLIST default-action-ref
    name CDATA #REQUIRED
>

<!ELEMENT external-ref (#PCDATA)>
<!ATTLIST external-ref
    name NMTOKEN #REQUIRED
    required (true|false) "true"
>

<!ELEMENT global-results (result+)>
```

```
<!ELEMENT global-exception-mappings (exception-mapping+)>

<!ELEMENT action (param|result|interceptor-ref|exception-mapping|external-ref)*>
<!ATTLIST action
    name CDATA #REQUIRED
    class CDATA #IMPLIED
    method CDATA #IMPLIED
    converter CDATA #IMPLIED
>

<!ELEMENT param (#PCDATA)>
<!ATTLIST param
    name CDATA #REQUIRED
>

<!ELEMENT result (#PCDATA|param)*>
<!ATTLIST result
    name CDATA #IMPLIED
    type CDATA #IMPLIED
>

<!ELEMENT exception-mapping (#PCDATA|param)*>
<!ATTLIST exception-mapping
    name CDATA #IMPLIED
    exception CDATA #REQUIRED
    result CDATA #REQUIRED
>

<!ELEMENT include (#PCDATA)>
<!ATTLIST include
    file CDATA #REQUIRED
>
```

xwork.xml的例子

```
<xwork>
    <include file="webwork-default.xml"/>

    <package name="person" extends="webwork-default" namespace="/person">
        <action name="listPeople" class="com.opensymphony.webwork.showcase.person.ListPeople">
            <interceptor-ref name="validationWorkflowStack"/>
            <result type="freemarker">listPeople.ftl</result>
        </action>

        <!-- our JasperReports example -->
        <action name="jasperList"
class="com.opensymphony.webwork.showcase.jasper.JasperAction">
            <result name="success" type="jasper">
                <param name="location">/jasper/sample_report.jasper</param>
                <param name="dataSource">people</param>
                <param name="documentName">peoplereport</param>
            </result>
        </action>


        <action name="newPerson" class="com.opensymphony.webwork.showcase.person.CreatePerson">
            <result type="redirect">listPeople.action</result>
            <result name="input" type="freemarker">newPerson.ftl</result>
        </action>

        <action name="editPerson" class="com.opensymphony.webwork.showcase.person.EditPerson">
            <result>editPeople.jsp</result>
        </action>

        <action name="doEditPerson" class="com.opensymphony.webwork.showcase.person.EditPerson"
method="save">
            <result name="error">editPeople.jsp</result>
            <result type="redirect">listPeople.action</result>
        </action>
    </package>
</xwork>
```

更多配置信息请查看XWork配置

Continuations

This page last changed on May 19, 2006 by scud.

Continuations是WebWork从RIFE项目借用的特性, 可以以极其简单的方式提供状态管理和类似向导的功能。(that allow for extremely simple state management and wizard-like functionality.)

Continuations现在还没有完成, 因此, 目前我们不推荐在重要的(heavy-use)产品开发中使用它. 我们将继续增强并稳定该特性，直到确信它能适用于极端的网站流量和用例.

设置

首先需要标识需要使用continuation支持的class所在的package(base package). 这需要在webwork.properties中使用属性 **webwork.continuations.package** 来标识. 通常, 可以设置顶级package名称(root package)如 **com.acme**. (这里的root package可以理解为package的前缀, 例如: 有两个package com.abc.p1和com.abc.p2, 那么只需要指定com.abc为webwork.continuations.package即可, 译注)

完成这一步骤后, WebWork会分析顶级package中的class并自动为使用continuation特性的class - 继承ActionSupport类并在 **execute()** 方法中调用 **pause()** 方法的class - 应用continuation支持.

URL Concerns (URL相关内容)

使用continuations特性必须由WebWork管理流程状态, 因此, 这要求应用程序必须将流程的ID告知WebWork. WebWork使用一个名为 **continue** 的参数为流程中的每一次请求提供一个唯一id来做到这一点. 如果使用URL或Form标签来产生URL链接, 这些标签会自动完成这一工作(是指加入 **continue** 参数, 译注). 如果没有使用这些标签, continuations不会正确工作.

Interceptor Concerns (截取器相关内容)

由于continuations从根本上改变了Action执行的方式, 因此理解它会对截取器带来那些影响是十分重要的. 最重要的一点是continuations只有在execute()方法被调用时才会发生作用(原文为kick in, 译注). 这意味着每一次请求(不管是新请求还是持续的请求)都会调用截取器. 这就是尽管Java调用栈中的其他内容(这里指局部变量等运行时数据, 译注)看起来是一样的, 但却能够在Action中取得新的请求参数值(apply new request parameters to your action)的原因.

通常这正是你想要的功能, 除了某些截取器, 比如Execute and Wait Interceptor, 可能还有Token Session Interceptor, 他们对Action调用的流程/周期有着截然不同的预期(have very different expectations about the workflow/lifecycle of the action invocation, ExecuteAndWait和Token都会限制同一个Session的请求次数, 也就是说他们不"期望"看到相同的请求, 译注). 在这些情况下, 不能使用continuations.

Example(示例)

使用continuations非常容易. 最大的问题是截然不同的应用流程的会话风格. 通常情况下, 你会使用session级别的变量或隐藏表单字段来传递状态. 而在使用continuations时, 使用Java语言本身来控制状态. 下面是继承ActionSupport的Guess类的主体代码:

```
public class Guess extends ActionSupport implements Preparable {
    int guess;
```

```
        public void prepare() throws Exception {
            // We clear the error message state before the action.
            // That is because with continuations, the original (or cloned) action is being
            //  executed, which will still have the old errors and potentially cause problems,
            //  such as with the workflow interceptor
            clearErrorsAndMessages();
        }

        public String execute() throws Exception {
            int answer = new Random().nextInt(100) + 1;
            int tries = 5;

            while (answer != guess && tries > 0) {
                pause(Action.SUCCESS);

                if (guess > answer) {
                    addFieldError("guess", "Too high!");
                } else if (guess < answer) {
                    addFieldError("guess", "Too low!");
                }

                tries--;
            }

            if (answer == guess) {
                addActionMessage("You got it!");
            } else {
                addActionMessage("You ran out of tries, the answer was " + answer);
            }

            return Action.SUCCESS;
        }

        public void setGuess(int guess) {
            this.guess = guess;
        }
    }
```

注意看class是如何将状态(本例中为tries)作为一个局部变量在execute()方法中保持的. WebWork的continuations特性能够在pause()方法执行后自动恢复(pick up)调用并将恢复(restore)全部局部变量, 就像循环"魔术般的"继续了一样(as if the logical loop is continuing "magically") (继续阅读以了解它是如何工作的).

Guess使用的视图没有什么特别的, 事实上他遵循 URL concerns 的要求使用Form标签来产生URL. 这将确保 continue 参数被包含在所有请求中.

```
<html>
<head>
    <title></title>
</head>

<body>
<#list actionMessages as msg>
    ${msg}
</#list>

<@ww.form action="guess" method="post">
    <@ww.textfield label="Guess" name="guess"/>
    <@ww.submit value="Guess"/>
</@ww.form>
</body>
</html>
```

Advanced: How it Works (进阶: 工作原理)

Continuations不是魔术, 虽然有时候看起来像是魔法. 事实上, 它使用某种非常智能的字节代码(byte-code)处理来完成工作. 这意味着 **为了使用continuations, 你的应用部署环境必须允许使用定制的class loader来加载活动类** (基于安全设置, 译注). 这通常不是问题, 但在这里必须说明.

当class被请求加载时, WebWork会把请求转交给RIFE/Continuations模块, 该模块将检查几个条件:

1. 该类继承了ActionSupport吗?
2. 该类定义了execute()方法吗?
3. 在execute()方法中, 调用了pause()方法吗?

如果三个条件的答案都是yes, 该类将被改动, execute()方法被使用try/catch, goto语句以及智能的"状态恢复"代码重写. 这些都是透明的不会影响调试或修改该类.

更多信息请参阅ActionSupport类pause()方法的JavaDocs文档:

> 立即中止action调用(通过抛出PauseException)并导致action调用返回指定的结果, 如#SUCCESS, #INPUT等.
>
> 当该action下一次被调用时(并且使用了相同的continue ID), 该方法将立即返回, 并将execute方法的整个调用栈恢复回来.
>
> 注意: 该方法 **只能** 在execute()方法中调用.

This page last changed on Mar 30, 2006 by scud.

The cookbook currently contains a lot of information that may be out of date. These pages will be updated over time and this warning will eventually be removed when the WebWork team feels that the content is 100% correct.

Webwork Cookbook

Welcome to the Webwork Cookbook. This page is geared towards providing an exchange of information for developers. Your welcome to share knowledge and any helpful tips here. Don't forget to check out the Jira (http://jira.opensymphony.com/browse/WW), which contains several contributions for WW which may not be listed here.

Setup

Deployment notes
App Servers
Setting up Eclipse with Tomcat
Using Maven to set up an Eclipse project for Webwork

Interceptors

Interceptor Order
File Upload Interceptor
Webwork file upload handling
Transparent web\-app I18N

Result examples

Redirect After Post Technique
JFreeChartResult
GroovyResult
RomeResult

Validation

How to validate field formats, such as a phone number

Ajax

Varia

Value Stack Internals
Using WebWork Components

OGNL
Accessing application, session, request objects
Application, Session, Request objects in jsp
Application, Session, Request objects in vm
How to format dates and numbers
Iterator tag examples
Tabular inputs with XWorkList
Exposing webwork objects to JSTL, with a JSTL and DisplayTag Example
Webwork 2 HTML form buttons Howto
Using Checkboxes
Using WebWork and XWork with JSP 2.0 and JSTL 1.1
Describing a bean in velocity
How do I populate a form bean and get the value using the taglib
Webwork 2 skinning

Access to Webwork objects from JSP 2.0 EL

This page last changed on Mar 30, 2006 by scud.

To access Webwork ValueStack from third party JSP taglibs you have to expose property values to JSP.

You can use Webwork2 tag <ww:set/> to set named parameter in a JSP page, request, session or application scope. Following example, sets a request scoped parameter 'a' to list of integers:

```
<ww:set name="'a'" value="{ 1, 2, 3, 4 }" scope="request"/>
```

After setting parameter, third party JSP taglibs can access variables, or you can use JSP 2.0 EL (Expression Language). This is convenient as short hand EL expression syntax
$

Unknown macro: {expression}
can be used in a text or inside of tag attributes:

```
a[0] = ${a[0]}

<sample:tag value="${a[1]}"/>
```

In practice, you've got to expose a lot of different variables to make effective use of third party taglibs like displaytag or wurfl. This leads to a lot of `<ww:set/>` tags what made me investigate how to make access to ValueStack and OGNL more transparent.

Why can't we just replace EL with OGNL?

Unfortunately, it isn't that simple. I've tinkered with `JSPFactory.setDefault()` to wrap around `getPageContext()` and create `ExpressionEvaluator` that would use OGNL.

This works in practice, but code generated by Jasper2 doesn't call `JSPFactory.getPageContext().getExpressionEvaluator()` but goes directly to static method that is hardwired to jakarta commons-el implementation.

Even if it would work it wouldn't be clean as `JSPFactory.setDefault()` should only be called by JSP implementation.

There is a simple, if not elegant, solution available in JSP 2.0 EL, for exposing ValueStack to OGNL. It is possible to create custom functions that can be called from EL expressions. Functions have to be 'public static' and specified in a TLD file.
Just import TLD in a JSP file where you've want to use a function.

For example, you could access action properties by evaluating OGNL expression by a function 'vs' (for valuestack) in EL:

```
<%@ taglib uri="/WEB-INF/tld/wwel.tld" prefix="x" %>

a[0] = ${x:vs('a[0]')}
a[0] * 4 = ${x:vs('a[0] * 4')}

Current action name: ${x:name()}
Top of ValueStack: ${x:top()}
```

To use this code you've got to add `wwel.tld` and `Functions.java` to your webapp project.

I would urge webworkers to define a set of functions that would be usable to wide community and include this in some future Webwork release.

```
<?xml version="1.0"?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/xml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
        version="2.0">

<description>
This taglib enables access to WebWork2 ValueStack
from JSP 2.0 Expression Language
</description>

<tlib-version>1.0</tlib-version>

<short-name>wwel</short-name>

<function>
        <name>vs</name>
        <function-class>com.nmote.wwel.Functions</function-class>
        <function-signature>
                java.lang.Object findOnValueStack(java.lang.String)
        </function-signature>
</function>

<function>
        <name>name</name>
        <function-class>com.nmote.wwel.Functions</function-class>
        <function-signature>
                java.lang.Object getActionName()
        </function-signature>
</function>

<function>
        <name>top</name>
        <function-class>com.nmote.wwel.Functions</function-class>
        <function-signature>
                java.lang.Object getTopOfValueStack()
        </function-signature>
</function>

</taglib>
```

```
package com.nmote.wwel;

import com.opensymphony.xwork.ActionContext;

/**
 * Utility functions for accessing webwork value stack and action context
 * from JSP 2.0 EL taglibs.
 *
 * @author Vjekoslav Nesek (vnesek@nmote.com)
 */
public class Functions {

        public static Object findOnValueStack(String expr) {
                ActionContext a = ActionContext.getContext();
                Object value = a.getValueStack().findValue(expr);
                return value;
        }

        public static Object getTopOfValueStack() {
                ActionContext a = ActionContext.getContext();
                Object value = a.getValueStack().peek();
                return value;
        }

        public static Object getActionName() {
                ActionContext a = ActionContext.getContext();
                Object value = a.getName();
                return value;
```

```
            }
    }
```

Accessing application, session, request objects

This page last changed on May 19, 2006 by scud.

Webwork provides several access helpers to access Session, Application, Request scopes.
Web agnostic (independent of the servlet API) with calls:

```
Map session = (Map) ActionContext.getContext().get("session");
session.put("myId",myProp);
```

The following gives you the same thing as above:

```
ServletActionContext.getRequest().getSession()
```

Note: Be sure not to use ActionContext.getContext() in the constructor of your action since the values may not be set up already (returning null for getSession()).
Note also: ActionContext.getContext().get("session") is the same as
ActionContext.getContext().getSession() with a cast to Map.

If you really need to get access to the HttpSession, use the ServletConfigInterceptor (see Interceptors).

In your views, you can access with your jsps as such

```
<ww: property value="#session.myId" />

<ww: property value="#request.myId" />
```

All the servlet scopes can be accessed like above.

```
Map request = (Map) ActionContext.getContext().get("request");
request.put("myId",myProp);
Map application = (Map) ActionContext.getContext().get("application");
application.put("myId",myProp);
Map session = (Map) ActionContext.getContext().get("session");
session.put("myId", myProp);
Map attr = (Map) ActionContext.getContext().get("attr");
attr.put("myId",myProp);
```

The 'attr' map will search the javax.servlet.jsp.PageContext for the specified key. If the PageContext dosen't exist, it will search request,session,application maps respectively.

Application, Session, Request objects in jsp

This page last changed on Mar 30, 2006 by scud.

The application, session and request objects are available from within ww tags in jsp wherever ognl can be evaluated. Use the #session syntax to get the object and access values by their keys using ['key'].

```
<ww:property value="#application\['foo'\]"/>

<ww:property value="#session\['baz'\]"/>
```

Conversely, if you would like to make webwork objects availible to say the jsp/jstl request scope. The property tag can be used like this.

```
<ww:set name="jobz" value="jobs" scope="request" />
```

A full example below shows a webwork variable "jobs" being exposed as "jobz" and being used with jstl and the display tag.

WW:Exposing webwork objects to JSTL, with a JSTL and DisplayTag Example

Application, Session, Request objects in vm

This page last changed on Mar 30, 2006 by scud.

```
$req.session.servletContext.getAttribute(...)
$req.session.getAttribute(...)
$req.getAttribute(...)
```

To get parameters from the QueryString or from a POSTed form, do not use getAttribute, use:

```
$req.getParameter(...)
```

But that's quite obvious, since $req is the request object and we all know how it works.

Example:

test.jsp:

```
<html><head></head><body>
<%
session.setAttribute("sessionFoo", "sessionBar");
session.getServletContext().setAttribute("applicationFoo", "applicationBar");
%>

<p>The following information should be available when sending the form below:

<ul>
        <li>Request parameter 'querystringFoo' with value 'querystringBar';</li>
        <li>Request parameter 'formFoo' with value 'formBar';</li>
        <li>Session attribute 'sessionFoo' with value 'sessionBar';</li>
        <li>Application attribute 'applicationFoo' with value 'applicationBar'.</li>
</ul>
</p>

<form action="test.vm?querystringFoo=querystringBar" method="post">
<input type="hidden" name="formFoo" value="formBar">
<p><input type="submit" value="Test!"></p>
</form>
</body></html>
```

test.vm:

```
<html><head></head><body>

#set ($ses = $req.getSession())
#set ($app = $ses.getServletContext())

<p>applicationFoo = $!app.getAttribute("applicationFoo")
<code>(app.getAttribute("applicationFoo"))</code></p>
<p>sessionFoo = $!ses.getAttribute("sessionFoo")
<code>(ses.getAttribute("sessionFoo"))</code></p>
<p>formFoo = $!req.getParameter("formFoo") <code>(req.getParameter("formFoo"))</code></p>
<p>querystringFoo = $!req.getParameter("querystringFoo")
<code>(req.getParameter("queryStringFoo"))</code></p>

</body></html>
```

Describing a bean in velocity

This page last changed on Mar 30, 2006 by scud.

The follow snippet might be useful during debugging to list the properties inside an
arbitary bean. Or for handing to a UI developer that use unaware of the getters/setters inside an object.

```
## prints out the property names for a bean
#macro (describeBean $name)
#set($bu = $webwork.bean("com.opensymphony.util.BeanUtils"))
        #foreach($propName in $bu.getPropertyNames($name))
                <li>$propName</li>
        #end
#end
```

i.e. assuming $obj is a PersonObject that has properties(firstName, lastName, and zip).

```
#describeBean($obj)
```

would print
<li>firstName</li>
<li>lastName</li>
<li>zip</li>

One might also expand upon this to build a dynamic interface with via reflection. e.g.

```
$webwork.evalute("$obj.${propName}")
```

Exposing webwork objects to JSTL, with a JSTL and DisplayTag Example

This page last changed on Mar 30, 2006 by scud.

```
<ww:set name="jobz" value="jobs" scope="request" />
```

The full example below shows a webwork variable "jobs" being exposed as "jobz" to the request scope and being used with jstl and the display tag.

```
<%@ taglib uri="/WEB-INF/tlds/c.tld" prefix="c" %>
<%@ taglib uri="/WEB-INF/tlds/fmt.tld" prefix="fmt" %>
<%@ taglib uri="/WEB-INF/tlds/displaytag-el-12.tld" prefix="display" %>
<%@ taglib uri="/WEB-INF/tlds/webwork.tld" prefix="ww" %>

<ww:set name="jobz" value="jobs" scope="request" />

<h1><fmt:message key="title.listAllJobs"/></h1>
<display:table name="jobz" class="simple" id="row" >
  <display:column  titleKey="label.global.actions" >
                <c:url var="viewurl" value="/viewJobDetail.action">
                        <c:param name="name" value="${row.name}"/>
                        <c:param name="groupName" value="${row.group}"/>
                </c:url>
                <c:url var="exeurl" value="/viewJobDetail.action">
                        <c:param name="name" value="${row.name}"/>
                        <c:param name="groupName" value="${row.group}"/>
                        <c:param name="executeJobAction" value="execute"/>
                </c:url>
                <c:url var="editurl" value="/viewJobDetail.action">
                        <c:param name="name" value="${row.name}"/>
                        <c:param name="groupName" value="${row.group}"/>
                        <c:param name="editAction" value="edit"/>
                </c:url>
        <a href='<c:out value="${viewurl}"/>'><fmt:message key="label.global.view"/></a> |
        <a href='<c:out value="${editurl}"/>'><fmt:message key="label.global.edit"/></a> |
        <a href='<c:out value="${exeurl}"/>'><fmt:message key="label.global.execute"/></a>
&nbsp;
  </display:column>

  <display:column property="group" titleKey="label.job.group" sortable="true"   />
  <display:column property="name" titleKey="label.job.name" sortable="true"  />
  <display:column property="description" titleKey="label.job.description" />
  <display:column property="jobClass" titleKey="label.job.jobClass" sortable="true"  />

</display:table>
```

Please note, at the time of this writing the "titleKey" attribute of the display tag's column tag is not yet released into a final version. It is a feature that is currently, only available through cvs.

GroovyResult

This page last changed on Mar 30, 2006 by scud.

GroovyResult - Groovy scripts as a view

This is an attempt to create a Result type that uses Groovy (http://groovy.codehaus.org) files as a view. It exposes the current ActionContext to a groovy script. This doesn't really have much practical use, but it's fun nonetheless and shows how easy creating Webwork Results is. There is another Result (JFreeChartResult) in the Cookbook

Installation

Not much - just make sure you have Groovy in your classpath, and the antlr, asm-* and groovy jars available to your webapp.

Configuration

xwork.xml - result-types definitions

```
<result-types>
   <result-type name="groovy" class="myapp.webwork.extensions.GroovyResult"/>
</result-types>
```

xwork.xml - action definitions

```
<action name="MyAction" class="myapp.webwork.actions.MyAction">
  <result name="success" type="groovy">
    <param name="file">test.groovy</param>
  </result>
</action>
```

The result type takes one parameter (for now), namely 'file', which contains the name of the groovy script in our script directory.

Show me the code !

Here's the code of the actual GroovyResult. This is a verbose version, with a lot of error checking.
GroovyResult.java - source code

```
public class GroovyResult implements Result {

        public final static String GROOVY_DIR_NAME = "groovy";

        private final static Logger logger = Logger.getLogger(GroovyResult.class);
        //our groovy source file name
 private String file;
        //a groovy shell
 private GroovyShell shell;
        //our parsed script
```

```
    private Script script;
           //the outputstream that will replace the 'out' in our groovy stream
    private OutputStream out;
           //directory containing groovy scripts
    private String scriptDirectory;
           /*
            * (non-Javadoc)
            *
            * @see com.opensymphony.xwork.Result#execute(com.opensymphony.xwork.ActionInvocation)
            */
           public void execute(ActionInvocation inv) {

                   //check the scriptDirectory - if it doesn't exists, use the default one
            //WEBAPP + Groovy files directory
            if (scriptDirectory == null) {
                           //not pretty, but this allows us to get the app root directory
                    String base = ServletActionContext.getServletContext().getRealPath(
                                            "/");
                           //if for some reason (.war, apache connector, ..) we can't get the
                    // base path
                    if (base == null) {
                                   logger
                                                   .warn("Could not translate the virtual path
   \"/\" to set the default groovy script directory");
                                   return;
                           }
                           scriptDirectory = base + GROOVY_DIR_NAME;
                           //issue a warning that this directory should NOT be world readable
                    // !!
                    logger
                                            .warn("Please make sure your script directory is NOT
   world readable !");
                   }

                   // first of all, make sure our groovy file exists, is readable, and is
            // an actual file

                   File groovyFile = new File(scriptDirectory, file);
                   if (!groovyFile.exists()) {
                           //log an error and return
                    logger.warn("Could not find destination groovy file: "
                                            + groovyFile.getAbsolutePath());
                           return;
                   }
                   if (!groovyFile.isFile()) {
                           //log an error and return
                    logger.warn("Destination is not a file: "
                                            + groovyFile.getAbsolutePath());
                           return;
                   }
                   if (!groovyFile.canRead()) {
                           //log an error and return
                    logger.warn("Can not read file: " + groovyFile.getAbsolutePath());
                           return;
                   }

                   if (logger.isDebugEnabled())
                           logger.debug("File " + groovyFile.getPath()
                                            + " found, going to parse it ..");

                   /*
                    * Here we create a Binding object which we populate with the webwork
                    * stack
                    */
                   Binding binding = new Binding();

                   binding.setVariable("context", ActionContext.getContext());

                   /*
                    * We replace the standard OutputStream with our own, in this case the
                    * OutputStream from our httpResponse
                    */
                   try {
                           //the out will be stored in an OutputStream
                    out = ServletActionContext.getResponse().getOutputStream();
                   } catch (IOException e1) {
                           logger.error("Could not open outputstream", e1);
                   }
```

```
                if (out != null){
                        binding.setVariable("out", out);
                }
                else {
                        logger
                                        .warn("OutputStream not available, using default
System.out instead");
                        binding.setVariable("out", System.out);
                }

                //create a new shell to parse and run our groovy file
         shell = new GroovyShell(binding);
                try {
                        //try to parse the script - the returned script could be cached for
                 //performance improvent
                 script = shell.parse(groovyFile);
                } catch (CompilationFailedException e) {
                        logger.error("Could not parse groovy script", e);
                        return;
                } catch (IOException e) {
                        logger.error("Error reading groovy script", e);
                        return;
                }
                //the binding is set, now run the script
         Object result = script.run();

                if (logger.isDebugEnabled()) {
                        logger.debug("Script " + groovyFile.getName()
                                        + " executed, and returned: " + result);
                }
                try {
                        out.flush();
                } catch (IOException e2) {
                        logger.error("Could not flush the outputstream", e2);
                }
        }

        /**
         * @return Returns the script.
         */
        public Script getScript() {
                return script;
        }
        /**
         * @param file
         *            The file to set.
         */
        public void setFile(String file) {
                this.file = file;
        }
        /**
         * @param out
         *            The out to set.
         */
        public void setOut(OutputStream out) {
                this.out = out;
        }
```

Explanation

The first part of the result is little more than:

- determining the script directory - defaults to MYWEBAPP/groovy/
- checking the file - make sure it exists, is readable, ..

⚠ Make sure the groovy scripts directory is not world readable !

The groovy part starts at:

```
Binding binding = new Binding();
binding.setVariable("context", ActionContext.getContext());
```

A Binding object allows us to 'bind' objects to a groovy script, so they can be used as variables. In this case, I took the ActionContext and exposed it as 'context'.

```
out = ServletActionContext.getResponse().getOutputStream();
...
binding.setVariable("out", out);
```

We also bind an OutputStream to the groovy script (as 'out') - it simply serves as a replacement for the standard System.out, so any printing goes directly to the http response outputstream.

```
shell = new GroovyShell(binding);
```

Next step; we create a GroovyShell, and pass our populated Binding to the constructor. Any script ran by this shell will have access to the passed variables (ActionContext and OutputStream).

```
script = shell.parse(groovyFile);
```

Before you can run a groovyFile, you need to parse it. Any syntax errors will be reported here - I also suggest adding a better error reporting in this case if you actually want to use this Result.
Upon successful parsing, a Script is returned (which could be cached if you want to increase performance) which will be run by our Shell.

```
Object result = script.run();
```

As a test, you might want to create a little 'groovy' script to test our Result.
test.groovy - a simple groovy script

```
for (item in context.contextMap){
        println "item: ${item}"
}
```

Place the test.groovy file in your groovy scripts directory. You should now see the result when you invoke MyAction.action in your browser.

Possible improvements are binding all objects on the stack so they become available to the groovy script, refactoring to an InputStream instead of a File, etc .. Comments welcome !

Handing IoC Components to Interceptors and Validators

This page last changed on Mar 30, 2006 by scud.

Reason for use

I have recently found the need for my Interceptors and Validators to be able to access Components - such as a Validators which is UserAware and checks the UserManager to see if the user exists. Or a Interceptor which is ApplicationAware and asks the ApplicationManager if it is setup yet - if not, then redirecting to a setup action instead.

Currently WebWork (at version 2.1.7) only supports component management of Action, but this can be changed quite easily - if you know where to look.

Extending the Object Factorys

WebWork uses a **com.opensymphony.xwork.ObjectFactory** object instance to generate the various objects that WebWork utilises - Validators, Interceptors, Actions, and Results for example. This is the object we are going to extend to add some of this functionality.

The methods **buildInterceptor** and **buildValidator** do what they say on the tin. I have overriden them to do the following:

```
public Interceptor buildInterceptor(InterceptorConfig ic, Map map) throws
ConfigurationException {
        Interceptor i = super.buildInterceptor(ic, map);
        cm.initializeObject(i);
        return i;
    }

    public Validator buildValidator(String string, Map map) throws Exception {
        Validator v = super.buildValidator(string, map);
        cm.initializeObject(v);
        return v;
    }
```

Creating a Component Mananger

The variable **cm** is a **ComponentManager**. As I am unsure of how to access the ComponentManager that is used in the ComponentInterceptor (or used when initalizing Action objects), we have to create our own. As the ObjectFactory is a singleton the overhead of this is relatively minor, even though not ideal.

The ComponentManager is created in the constructor like this:

```
private static final Log log = LogFactory.getLog(ObjectFactory.class);

    private ComponentConfiguration cc;
    private ComponentManager cm;

    public ObjectFactory() {
        super();
        cm = (ComponentManager) ActionContext.getContext().get(
ComponentInterceptor.COMPONENT_MANAGER );

        if (cm == null) {
            cc = new ComponentConfiguration();
            InputStream configXml =
Thread.currentThread().getContextClassLoader().getResourceAsStream("components.xml");
            try {
                cc.loadFromXml(configXml);
            } catch (Exception e) {
                log.info("No component.xml found. They test will continue without initializing
components.");
                cc = null;
            }

            cm = new DefaultComponentManager();
            if (cc != null) {
                cc.configure(cm, "session");
                cc.configure(cm, "application");
                cc.configure(cm, "request");
            }
        }
    }
```

Using our new ObjectFactory

The ObjectFactory is a singleton which allows you to set the object it hands out. To do this I have chosen to override the `init` method of the `com.opensymphony.webwork.dispatcher.ServletDispatcher` class. The method looks something like this:

```
public void init(ServletConfig servletConfig) throws ServletException {
        ObjectFactory.setObjectFactory( new planb.jobsite.xwork.ObjectFactory() );
        super.init(servletConfig);
    }
```

Code Results

The following full files result from this article.

Object Factory

```
import com.opensymphony.xwork.interceptor.Interceptor;
import com.opensymphony.xwork.interceptor.component.ComponentManager;
import com.opensymphony.xwork.interceptor.component.DefaultComponentManager;
import com.opensymphony.xwork.interceptor.component.ComponentInterceptor;
import com.opensymphony.xwork.interceptor.component.ComponentConfiguration;
import com.opensymphony.xwork.config.entities.InterceptorConfig;
import com.opensymphony.xwork.config.ConfigurationException;
import com.opensymphony.xwork.validator.Validator;
import com.opensymphony.xwork.ActionContext;

import java.util.Map;
import java.io.InputStream;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

public class ObjectFactory extends com.opensymphony.xwork.ObjectFactory {

    private static final Log log = LogFactory.getLog(ObjectFactory.class);

    private ComponentConfiguration cc;
    private ComponentManager cm;

    public ObjectFactory() {
        super();
        cm = (ComponentManager) ActionContext.getContext().get(
ComponentInterceptor.COMPONENT_MANAGER );

        if (cm == null) {
            cc = new ComponentConfiguration();
            InputStream configXml =
Thread.currentThread().getContextClassLoader().getResourceAsStream("components.xml");
            try {
                cc.loadFromXml(configXml);
            } catch (Exception e) {
                log.info("No component.xml found. They test will continue without initializing
components.");
                cc = null;
            }

            cm = new DefaultComponentManager();
            if (cc != null) {
                cc.configure(cm, "session");
                cc.configure(cm, "application");
                cc.configure(cm, "request");
            }
        }
    }

    public Interceptor buildInterceptor(InterceptorConfig ic, Map map) throws
ConfigurationException {
        Interceptor i = super.buildInterceptor(ic, map);
        cm.initializeObject(i);
        return i;
    }

    public Validator buildValidator(String string, Map map) throws Exception {
        Validator v = super.buildValidator(string, map);
        cm.initializeObject(v);
        return v;
    }

}
```

Servlet Dispatcher

```
import com.opensymphony.xwork.ObjectFactory;

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;

public class ServletDispatcher extends com.opensymphony.webwork.dispatcher.ServletDispatcher {

    public void init(ServletConfig servletConfig) throws ServletException {
        ObjectFactory.setObjectFactory( new planb.jobsite.xwork.ObjectFactory() );
        super.init(servletConfig);
    }

}
```

web.xml

Replace the reference to the webwork ServletDispatcher to point to the above ServletDispatcher class.

Important Notes

You should find your Interceptors and Validators are now componentized just like Actions, however there are some important notes to be made.

Lifecycle Issues

Interceptors and Validators are both cached by webwork and reused instead of being reinstanciated - this will mean that you may experiance issues with components outside of the application scope. As all of my Interceptor / Validator required components are in this scope, this isn't an issue to me.

One solution to this constraint would be to investigate how webwork caches its Interceptors and Validators, then check to see if the objects use session / request scoped components and cache accordingly. Maybe a thought for the guys planning the next release of webwork!

Conclusion

For now this concludes this article - feel free to add your ideas!

How do I populate a form bean and get the value using the taglib

This page last changed on Mar 30, 2006 by scud.

First off, if you're coming from Struts, you may feel more comfortable using FormBeans instead of using the Action as your form bean. Be aware, though, that in Webwork you DO have the option of having the properties directly in the Action class. If you want to use a FormBean, here's an example:

```
public class TestAction extends ActionSupport {
    private TestBean myBean;

    public TestBean getMyBean() {
        return myBean;
    }

    public void setMyBean(TestBean myBean) {
        this.myBean = myBean;
    }

    protected String doExecute() throws Exception {
        myBean = new TestBean();
        BeanUtil.setProperties(ActionContext.getContext().getParameters(), myBean);
        return SUCCESS;
    }
}
```

Then, in your success.jsp, which is mapped as the success result of TestAction in the views.properties or actions.xml (see the docs for how to configure actions and view mappings), you can do this:

```
<!-- This will call getMyBean() on your action and put it on the top of the value stack ->
<webwork:property value="myBean">
<!- This will call getName() on your TestBean and print it to the page -->
The name is: <webwork:property value="name"/>
</webwork:property>
```

This is a good way to do it if you have several parameters from the TestBean that you want to display, but, if you have just one, like in this case, it's probably better to do this:

```
<webwork:property value="myBean/name"/>
```

NOTE:
As of WW2.2, the following should be used

```
<webwork:property value="myBean.name" />
```

Which will call getMyBean.getName() and print that out to the page.

How to format dates and numbers

This page last changed on Mar 30, 2006 by scud.

A frequently asked question is how best to display dates and numbers using a specified format. There are a number of approaches for this, the most naive of which would be to add a method to your action class to do the formatting for you. This method would take in a Date (or subclass) object as a parameter, and return a formatted String.

That approach however suffers from a number of flaws. For example, it is not i18n aware. The date format specified is rigid, and will not adapt to different locales easily (assuming you're not using a default formatter that is). It also clutters up your actions with code that has nothing to do with the action itself.

Instead, the recommended approach is to use Java's built-in date formatting features via use of the webwork:text tag.

The webwork:text tag should be used for all i18n values. It will look up the properties file for your action, and from that select the value for the key that you specify. This is best illustrated in an example:

```
<!-- display the number of items in a cart -->
<webwork:text name="'cart.items'" value0="cartItems" />
```

The above tag will work as follows. value0 will result in a call to **getCartItems()** on your action class. The **cart.items** name is escaped, so it is treated as a literal key into the actions' properties file. Your MyAction.properties file will contain the following:

cart.items=You have {0} items in your cart.

Normal Java **MessageFormat** behaviour will correctly substitute {0} with the value obtained from getCartItems.

Needless to say, this can get a lot more elaborate, with the ability to specify both date and number formatting. Let us consider another example. The goal here is to display a greeting to the user, as well as the date of their last visit.

```
<webwork:text name="'last.visit'" value0="userName" value1="lastVisit(userName)" />
```

MyAction.java contains:

```
public String getUserName() { ... };
public Date getLastVisit(String userName) { ... };
```

Your **MyAction.properties** file will then contain:

last.visit=Welcome back {0}, your last visit was at {1,date,HH:mm dd-MM-yyyy}

As you can see, this is a very powerful mechanism and allows you to easily display numbers and dates using any formatting rules that Java allows.

value0 interface deprecated

The examples above pass in the values as:

```
<webwork:text name="'text.message'" value0="userName"/>
```

These values should now (>2.1.7?) be passed as params:

```
<webwork:text name="'text.message'">
    <webwork:param value="'userName'"/>
</webwork:text>
```

Some message format examples

Here are some examples of formatting in the properties file:

```
format.date = {0,date,MM/dd/yy}
format.time = {0,date,MM/dd/yy ha}
format.percent = {0,number,##0.00'%'}
format.money = {0,number,$##0.00}
```

How to validate field formats, such as a phone number

This page last changed on Mar 30, 2006 by scud.

Validating the format of String fields for patterns (such as a phone number) is easy with StringRegexValidator (named "regex" in the default validator configuration).

Simply add the validator the field in question, and supply a regular expression to match it against.

```
<validators>
    <field name="phone">
        <field-validator type="regex">
            <param name="regex">\([\d][\d][\d]\) [\d][\d][\d]-[\d][\d][\d][\d]</param>
            <message>Phone number must be in the format (XXX) XXX-XXXX</message>
        </field-validator>
    </field>
</validators>
```

If your expression tests against alpha characters, you may be interested in the "caseSensitive" parameter of with Validator as well. It defaults to "true".

Interceptor Order

This page last changed on Mar 30, 2006 by scud.

Interceptors provide an excellent means to wrap before/after processing. The concept reduces code duplication (think AOP).

Order of interceptors...

```
<interceptor-stack name="xaStack">
  <interceptor-ref name="thisWillRunFirstInterceptor"/>
  <interceptor-ref name="thisWillRunNextInterceptor"/>
  <interceptor-ref name="followedByThisInterceptor"/>
  <interceptor-ref name="thisWillRunLastInterceptor"/>
</interceptor-stack>
```

Note that some interceptors will interrupt the stack/chain/flow... so the order is very important.

Iterceptors implementing com.opensymphony.xwork.interceptor.PreResultListener will run after the Action executes its action method but before the Result executes

```
thisWillRunFirstInterceptor
  thisWillRunNextInterceptor
    followedByThisInterceptor
      thisWillRunLastInterceptor
        MyAction1
        MyAction2 (chain)
        MyPreResultListener
        MyResult (result)
      thisWillRunLastInterceptor
    followedByThisInterceptor
  thisWillRunNextInterceptor
thisWillRunFirstInterceptor
```

Iterator tag examples

This page last changed on Mar 30, 2006 by scud.

This follows on from Iteration Tags which you should read first, but beware of references to '0' and 'that'; what you really want in WW2 is 'top', as illustrated below. (I finally worked this out from the source code - hopefully this page means you won't have to.)

Referencing the current value

The simple examples print out values from the list using the property tag, which uses the value at the top of the stack by default:

```
Days:
<ul>
<ww:iterator value="days">
    <li><ww:property/>
</ww:iterator>
</ul>
```

But if you're doing anything other than print the value, you probably need to refer to it specifically. Do this:

```
Most days:
<ul>
<ww:iterator value="days">
    <ww:if test="top != 'Monday'">
        <li><ww:property/>
    </ww:if>
</ww:iterator>
</ul>
```

Iterating over a list of objects

```
<ww:iterator value="employees">
    <ww:property value="name"/> is the <ww:property value="jobTitle"/><br>
</ww:iterator>
```

For 'name' and 'jobTitle' you could be more explicit and write 'top.name' and 'top.jobTitle', as 'top' refers to the object on the top of the stack. It's not necessary here, but it is in the next example.

Iterating over a list of lists

```
<table>
    <ww:iterator value="grid">
        <tr>
        <ww:iterator value="top">
            <td><ww:property/></td>
        </ww:iterator>
        </tr>
    </ww:iterator>
</table>
```

The trick here is to use 'top' as the value for the inner iterator. This example probably uses a two-dimensional array, but you can use the pattern for any list of lists.

A more complex example

In this example, 'countries' is a list of country objects, each of which has a name and a list of cities. Each city has a name.

```
<ww:iterator value="countries">
    <ww:iterator value="cities">
        <ww:property value="name"/>, <ww:property value="[1].name"/><br>
    </ww:iterator>
</ww:iterator>
```

The output looks like

```
Wellington, New Zealand
Auckland, New Zealand
Moscow, Russia
Glasgow, Scotland
Edinburgh, Scotland
Stockholm, Sweden
```

Both the country and city objects have a 'name' property. As you'd expect, the reference to 'name' on its own gives you the city name. To access the country name - effectively "hidden" by the city name - we refer to a specific position on the stack: '1'. The top of the stack, position 0, contains the current city, pushed on by the inner iterator; position 1 contains the current country, pushed there by the outer iterator.

Actually, as Patrick points out in his comment on Iteration Tags, the 'n' notation refers to a sub-stack beginning at position n, not just the object at position n. Thus '0' is the whole stack and '1' is everything except the top object. In our example, we could have been more specific about getting the country name and said '1.top.name'.

Misc

If no value is specified, iterator will try to grap object from the 'top' of the stack. If it is not iterable, ClassCastException will be thrown in the process. @see com.opensymphony.webwork.views.jsp.IteratorTag#doStartTag

JFreeChartResult

This page last changed on Mar 30, 2006 by scud.

Intro

I am rendering a chart to the output stream. Instead of streaming it directly to the response.out, I create a ChartResult, and let webwork do the chaining for me.

I generate the chart in one class, and I render it out in another class, effectively decoupling the view from the actions. You can easily render it out to a file or some view other than a web response.out if you wish.

Configuration

xwork.xml - result-types definitions

```
<result-types>
   <result-type name="chart" class="myapp.webwork.extensions.ChartResult"/>
</result-types>
```

xwork.xml - action definitions

```
<action name="viewModerationChart" class="myapp.webwork.actions.ViewModerationChartAction">
  <result name="success" type="chart">
    <param name="width">400</param>
    <param name="height">300</param> </result>
</action>
```

Source Codes

My result class searches for a "chart" in the ValueStack and renders it out...

```
public class ChartResult implements Result {

        private int width;
        private int height;

        public void execute(ActionInvocation invocation) throws Exception {
                JFreeChart chart =
                        (JFreeChart) invocation.getStack().findValue("chart");
                HttpServletResponse response = ServletActionContext.getResponse();
                OutputStream os = response.getOutputStream();
                ChartUtilities.writeChartAsPNG(os, chart, width, height);
                os.flush();
        }

        public void setHeight(int height) {
                this.height = height;
        }

        public void setWidth(int width) {
                this.width = width;
        }
}
```

My action class creates the JFreeChart to render...

```
public class ViewModerationChartAction extends ActionSupport {

        private JFreeChart chart;

        public String execute() throws Exception {
                // chart creation logic...
         XYSeries dataSeries = new XYSeries(new Integer(1)); //pass a key for this serie
         for (int i = 0; i <= 100; i++) {
                        dataSeries.add(i, RandomUtils.nextInt());
                }
                XYSeriesCollection xyDataset = new XYSeriesCollection(dataSeries);

                ValueAxis xAxis = new NumberAxis("Raw Marks");
                ValueAxis yAxis = new NumberAxis("Moderated Marks");

                // set my chart variable
         chart =
                        new JFreeChart(
                                "Moderation Function",
                                JFreeChart.DEFAULT_TITLE_FONT,
                                new XYPlot(
                                        xyDataset,
                                        xAxis,
                                        yAxis,
                                        new
StandardXYItemRenderer(StandardXYItemRenderer.LINES)),
                                false);
                chart.setBackgroundPaint(java.awt.Color.white);

                return super.SUCCESS;
        }

        public JFreeChart getChart() {
                return chart;
        }

}
```

Explaination

```
public JFreeChart getChart() {
        return chart;
}
```

makes the chart available on the ValueStack, which the result gets via

```
JFreeChart chart = (JFreeChart) invocation.getStack().findValue("chart");
```

From what I can deduce, the webwork pulls in the height and width variables from the xwork.xml definitions for that particular action...

```
<param name="width">400</param>
<param name="height">300</param>
```

Suggestions for the next developer...

Currently the "chart" property is hardcoded. There should be a better way of transferring data from the Action to the Result, via some externally defined variable or something.

As mentioned by John Patterson (mailing list), the Action is still dependant on a JFreeChart Chart class. This can be improved. The seperation between Action and View can be made cleaner. A chart-agonistic List or Array can be used as the data, and the configuration of the chart details (font, axis, etc...) be done via the result properties in the xwork.xml.

But hey, the above works for now. Any suggestions are welcome.

Creating charts via CeWolf directly in Velocity templates

See WW:CeWolf charts using Velocity templates.

redirect after post

This page last changed on Mar 30, 2006 by scud.

The redirect-after-post technique is a common pattern in web application development. It simply means making your action, after it has successfully executed, result in a redirect. You can do this by using the Redirect Result or the Redirect Action Result.

RomeResult

This page last changed on Mar 30, 2006 by scud.

Introduction

A couple of days ago I quickly had to create a WW result type that would transform a Rome (https://rome.dev.java.net/) SyndFeed to several news feeds. WW makes this very, very easy.

Used versions

WebWork 2.2 beta 4
Rome 0.7 beta

The code

```
/**
 *
 */
package com.acme.result;

import java.io.Writer;

import org.apache.log4j.Logger;

import com.opensymphony.webwork.ServletActionContext;
import com.opensymphony.xwork.ActionInvocation;
import com.opensymphony.xwork.Result;
import com.sun.syndication.feed.synd.SyndFeed;
import com.sun.syndication.io.SyndFeedOutput;

/**
 * A simple Result to output a Rome SyndFeed object into a newsfeed.
 * @author Philip Luppens
 *
 */
public class RomeResult implements Result {

        private String feedName;

        private String feedType;

        private final static Logger logger = Logger.getLogger(RomeResult.class);

        /*
         * (non-Javadoc)
         *
         * @see com.opensymphony.xwork.Result#execute(com.opensymphony.xwork.ActionInvocation)
         */
        public void execute(ActionInvocation ai) throws Exception {
                if (feedName == null) {
                        // ack, we need this to find the feed on the stack
                 logger
                                        .error("Required parameter 'feedName' not found. "
                                                        + "Make sure you have the param tag set
and "
                                                        + "the static-parameters interceptor
enabled in your interceptor stack.");
                        // no point in continuing ..
                 return;
                }

                // don't forget to set the content to the correct mimetype
         ServletActionContext.getResponse().setContentType("text/xml");
                // get the feed from the stack that can be found by the feedName
         SyndFeed feed = (SyndFeed) ai.getStack().findValue(feedName);

                if (logger.isDebugEnabled()) {
                        logger.debug("Found object on stack with name '" + feedName + "': "
                                        + feed);
                }
                if (feed != null) {

                        if (feedType != null) {
                                // Accepted types are: rss_0.90 - rss_2.0 and atom_0.3
                         // There is a bug though in the rss 2.0 generator when it checks
                         // for the type attribute in the description element. It's has a
                         // big 'FIXME' next to it (v. 0.7beta).
                         feed.setFeedType(feedType);
                        }
                        SyndFeedOutput output = new SyndFeedOutput();
                        //we'll need the writer since Rome doesn't support writing to an
outputStream yet
                 Writer out = null;
                        try {
                                out = ServletActionContext.getResponse().getWriter();
                                output.output(feed, out);
                        } catch (Exception e) {
                                // Woops, couldn't write the feed ?
                         logger.error("Could not write the feed", e);
                        } finally {
                                //close the output writer (will flush automatically)
                         if (out != null) {
                                        out.close();
                                }
                        }

                } else {
                        // woops .. no object found on the stack with that name ?
                 logger.error("Did not find object on stack with name '" + feedName
                                        + "'");
                }
        }

        public void setFeedName(String feedName) {
                this.feedName = feedName;
        }

        public void setFeedType(String feedType) {
                this.feedType = feedType;
        }

}
```

Code Explanation

Easy enough. We try to find the SyndFeed object on the WW stack. If we can find it, we will set the feed type (rss v0.9 +, atom 0.3) if it has been specified in the result parameters (see below). Then we use a SyndFeedOutput to write our newsfeed to our PrintWriter from our response.

XWork configuration

Before you can use this result, you will need to register it in your xwork.xml:

```
<package name="default" extends="webwork-default">
        <result-types>
                <result-type name="feed" class="com.acme.result.RomeResult"/>
        </result-types>
        <interceptors>
..
```

You can now use this result type. So, create an Action that will create and populate the Rome SyndFeed, and make sure you provide a getter for your populated SyndFeed. The actual creation of your feed is beyond this recipe, but you can find plenty of examples in the Rome Wiki (http://wiki.java.net/bin/view/Javawsxml/Rome05Tutorials).

```
<action name="feed" class="com.acme.action.CreateFeedAction">
        <result name="success" type="feed">
                <param name="feedName">feed</param>
                <param name="feedType">rss_1.0</param>
        </result>
</action>
```

Concluding thoughts

This is a simple feed result using Rome as a news feed generator. You might want to make sure you don't generate your feed on every request, but there are lots of ways to deal with such problems. You can also provide additional setters in the Result to set your feed title, url, etc, but this should suffice for a quickstart.

This page last changed on Mar 30, 2006 by scud.

Setting up Eclipse with Tomcat

Introduction

When I started using WebWork I went through the trouble of setting up my development environment to cater for web development. What I wanted was a fast turn around time, i.e. if I changed a .jsp file I could just refresh the web browser to see the changes. Also changes in WebWork related configurations will have immediate effect. And finally minor java changes from within Eclipse is also hot replaced.

I will document how to setup Eclipse and install a tomcat plugin so all is integrated within Eclipse and you can easily start, stop, restart Tomcat, see output in the Eclipse console, debug, make code changes on the fly, etc. etc. so development is a joy.

If you are fortunate enough to use the JetBrains IDEA editor then all this is already setup for you.

Preface

I used these tools

Windows XP
Eclipse 3.11
Tomcat 5.5.x
Sysdeo Tomcat Plugin Launcher 0.80

Install Tomcat Plugin

There are several Tomcat plugins to Eclipse, and I tried several versions. I found that Sysdeo was the best one for my needs.

There is a good Eclipse plugin homepage at http://www.eclipse-plugins.info

Download the Sysdeo plugin from it's homepage at http://www.sysdeo.com/eclipse/tomcatplugin

Unzip the downloaded file to <ECLIPSE_HOME>\plugins

Installing Tomcat

If you don't have Tomcat installed then download it from http://tomcat.apache.org

Configure Tomcat Plugin in Eclipse

Restart/Start Eclips so the plugin is loaded.

If the plugin is working you will immediately notice 3 new icons in the Eclipse toolbar

These icons are for starting, stopping and restarting Tomcat.
Now you need to configure the plugin **Windows** -> **Preference**. And the select the **Tomcat** tab.
Here you select the Tomcat version to 5.x and enter the <TOMCAT_HOME> folder. You leave Context decleration mode to Server.xml.
Leave the **advanced** tab as default. **JVM Settings** should have a JRE selected. And the **source path** should have ticked Automatically compute source path.

Create a new Project

When you create a new project in Eclipse you folders follow a certain structure standard to make this work.

I have this folder structure

```
<project_home>
+ src
   + java
+ webapp
   + WEB-INF
      + lib
```

In the **lib** folder copy your needed .jar files. I have these files:

```
commons-logging.jar
dwr.jar
freemarker.jar
javamail.jar
log4j-1.2.9.jar
ognl.jar
oscore.jar
rife-continuations.jar
servletapi.jar
spring.jar
webwork-2.2.1.jar
xwork.jar
```

And in WEB-INF your should have web.xml and the Spring configuration file. And if you use Spring log4j ConfigListener you could also store log4j.properties here. I have these 3 files:
applicationContext.xml
log4j.properties
web.xml

The spring configuration file is basically empty to start with **ApplicationContext.xml**

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN//EN"
"http://www.springframework.org/dtd/spring-beans.dtd">
<beans default-autowire="autodetect">
</beans>
```

And the web.xml

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>

        <context-param>
                <param-name>log4jConfigLocation</param-name>
                <param-value>/WEB-INF/log4j.properties</param-value>
        </context-param>

        <filter>
                <filter-name>webwork</filter-name>
<filter-class>com.opensymphony.webwork.dispatcher.FilterDispatcher</filter-class>
        </filter>

        <filter-mapping>
                <filter-name>webwork</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>

        <listener>
<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
        </listener>

        <listener>
<listener-class>org.springframework.web.util.Log4jConfigListener</listener-class>
        </listener>

        <servlet>
                <servlet-name>freemarker</servlet-name>
<servlet-class>com.opensymphony.webwork.views.freemarker.FreemarkerServlet</servlet-class>
        </servlet>

        <servlet-mapping>
                <servlet-name>freemarker</servlet-name>
                <url-pattern>*.ftl</url-pattern>
        </servlet-mapping>

        <welcome-file-list>
                <welcome-file>index.jsp</welcome-file>
                <welcome-file>index.html</welcome-file>
        </welcome-file-list>

</web-app>
```

And my log4j.properties

```
log4j.rootLogger=INFO, console

log4j.appender.console=org.apache.log4j.ConsoleAppender
log4j.appender.console.layout=org.apache.log4j.PatternLayout
log4j.appender.console.layout.ConversionPattern=%d{dd MMM yyyy HH:mm:ss} %-5p %c - %m%n

#log4j.category.org.springframework=DEBUG
#log4j.category.com.opensymphony.webwork=DEBUG
#log4j.category.com.opensymphony.xwork=DEBUG
```

Now the tricky part is that we still need some configuration files for WebWork that usually resides in the classpath folder. These files we must store outside the webapp folder due Eclipse hot java code deployer.

So these files:
   webwork.properties
   xwork.xml
Is stored in the **src/java** folder.

This is my **webwork.properties** that is setup for development

```
webwork.objectFactory = spring
webwork.devMode = true
webwork.configuration.xml.reload = true
webwork.url.http.port = 8080
```

And then the **xwork.xml** where we configure our action

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.1.1//EN"
"http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">
<xwork>
<include file="webwork-default.xml"/>
<package name="default" extends="webwork-default">
</package>
</xwork>
```

Now we are nearly ready. We must tell Eclipse to output the compiled source files to **webapp/WEB-INF/classes**. This is done by **Project** -> **Properties**. Then select the **Java Build Path** tab. Then type "<project_name>**webapp/WEB-INF/classes**" in the default output folder. Then Eclipse would put the compiled source files to our web app classes folder **AND** also copy all other configuration files within the **src/java** folder, and thus also our **webwork.properties** and **xwork.xml** is copied.

It is important to put **webwork.properties** and **xwork.xml** in the **src/java** folder as Eclipse will automatically clean the build source folder when it compiles. So if we put **webwork.properties** and **xwork.xml** in the **WEB-INF/classes** folder Eclipse will delete them. That is why we put the files in **src/java** instead.

Now we are done configuring and setup our development environment in Eclipse. Now let's test it.

Now let's make the famous Hello World and se code changes on the fly 🙂

In the webapp folder we could create a welcome page to see if tomcat works. So we create a **index.html** file

```
<html>
<body>
<h1>Hello World</h1>
<p/>
</body>
</html>
```

Now we are readt to start Tomcat so click the icon **Start Tomcat**. If everyting works you will se Tomcat startup and output in the console panel in Eclipse. Now point your webbrowser to the url http://localhost:8080. And the welcome page should be displayed.

Next we create a simple action named HelloAction

```
package dk.claus;

import com.opensymphony.xwork.ActionSupport;

public class HelloAction extends ActionSupport {

        private String world;

        public String getWorld() {
                return world;
        }

        public void setWorld(String world) {
                this.world = world;
        }

        public String execute() throws Exception {
                world = "Hello World from your Action";
                return SUCCESS;
        }

}
```

So we must change our xwork.properties to know this action

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.1.1//EN"
"http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">

<xwork>
    <include file="webwork-default.xml"/>

    <package name="default" extends="webwork-default">

        <action name="hello" class="dk.claus.HelloAction">
                <result name="success">hello.jsp</result>
        </action>

    </package>

</xwork>
```

And then we need to have a link on our welcome page, so we change it:

```
<html>
<body>
<h1>Hello World</h1>
<p/>

<a xhref="hello.action">Hello World Action</a>

</body>
</html>
```

And finally we must make a result page for the action, so we create a hello.jsp page.

```
<%@ taglib uri="/webwork" prefix="ww"%>

<html>
<body>
  <h1>This is hello world action</h1>
  <p/>
  What did the action say? <ww:property value="world"/>

</body>
</html>
```

Now save all these files and refresh your browser. The welcome page should now contain the link. Clicking the link would execute your action and show the result page.

Now change the code in the action by changing the world text to something different. Save the file and refresh the browser. Isn't this great

Hope this guide can help you setup a development environment that is trouly a joy to work with.

Stopping Tomcat

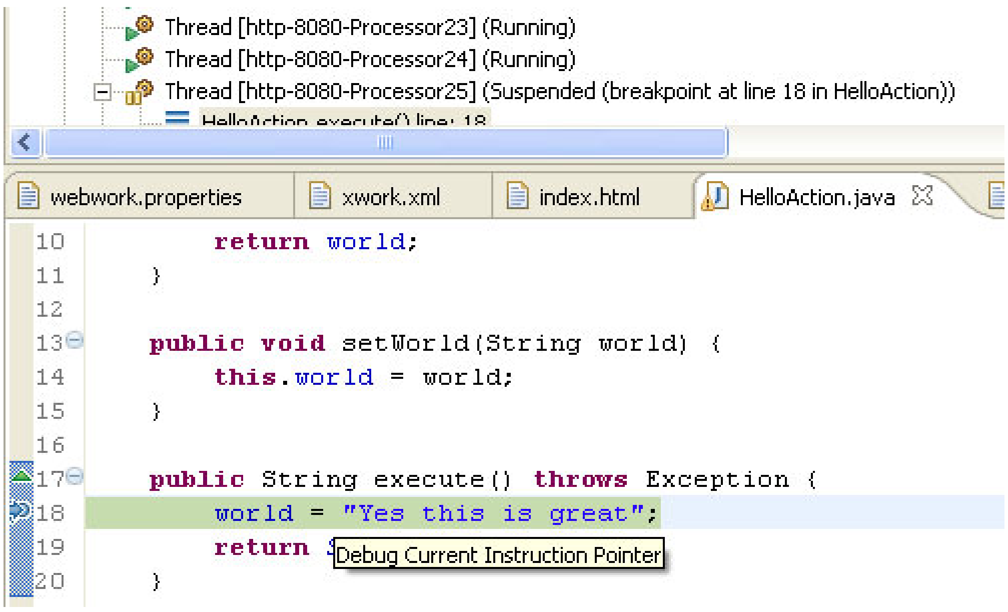
You must remember to use the 'Stop Tomcat' toolbar button. If you kill the application using the red terminate button in Eclipse you could potentially have Eclipse stop responding. This happened for me twice.

Hot code replace failed

If you add, rename or delete methods Eclipse can't replace the code and you will get a warning dialog. Here you should remember to click 'Continue' and then click the Stop Tomcat button (or Restart Tomcat button). This ensure that Tomcat is properly shutdown.

Debugging

Not a problem at all. Just set a breakpoint in your code and you are off to go. That is truly a joy and that can't be easiler.



```
Thread [http-8080-Processor23] (Running)
Thread [http-8080-Processor24] (Running)
Thread [http-8080-Processor25] (Suspended (breakpoint at line 18 in HelloAction))
    HelloAction.execute() line: 18
```

```java
        return world;
    }

    public void setWorld(String world) {
        this.world = world;
    }

    public String execute() throws Exception {
        world = "Yes this is great";
        return S
    }
```

The end

I hope this guide added something to the table. I decided I wanted to create this guide on this wiki site to contribute something back to this great web framework.

Tabular inputs with XWorkList

This page last changed on Mar 30, 2006 by scud.

Sometimes you need a way to enter tabular data such as list of quantity for products in a shopping cart, marks from a list of examination candiates, etc. If you just have one input value per line item, you can use a HashMap to store the value. This can be expanded to support multiple input values by having multiple HashMap. This describes a number of alternatives using some of more advanced features of WebWork. Assume you want to capture the quantity and a gift note for a list of products in a shopping cart (i.e Amazon).

1. When the number of line items is known

If you are using JSP:
the cart.jsp file in altSyntax

```
<ww:iterator value="cart.items">
  <ww:hidden name="cart.items[%{#rowstatus.index}].productId" value="%{productId}">
  <ww:textfield name="cart.items[%{#rowstatus.index}].qty" value="%{qty}" />
  <ww:textfield name="cart.items[%{#rowstatus.index}].note" value="%{note}" />
</ww:iterator>
```

the cart.jsp file (non altSyntax)

```
<ww:iterator value="cart.items">
  <ww:hidden name="'cart.items[' + #rowstatus.index + '].productId'" value="productId">
  <ww:textfield name="'cart.items[' + #rowstatus.index + '].qty'" value="qty" />
  <ww:textfield name="'cart.items[' + #rowstatus.index + '].note'" value="note" />
</ww:iterator>
```

Alternatively, if you use Velocity as your view technology of choice:
the cart.vm file

```
#foreach ( $item in $cart.items )
  #set($index = $velocityCount - 1)
  <input type="hidden" name="cart.items[$index].productId" value="$item.productId">
  <input type="text" name="cart.items[$index].qty" value="$item.qty">
  <input type="text" name="cart.items[$index].note" value="$item.note">
#end
```

the UpdateCartAction.class

```
public class UpdateCartAction extends ActionSupport {

        public Cart getCart() {
                // Lazy initialization
                Cart result = ActionContext.getContext().getSession().get("cart.key");
                if ( result == null ) {
                        result = new Cart();
                        ActionContext.getContext().getSession().put("cart.key", result);
                }
                return result;
        }

        public String execute() throws Exception {
                // Just ensuring our cart is initialized...
         Cart cart = getCart();

                // loop through a
```

```
    }
  }
```

the Cart.class

```
  public class Cart implements Serializable {
    private List items = new ArrayList();

    public List getItems() {
      return items;
    }

    public void addItem(CartItem item) {
        ...
    }
  }
```

the CartItem.class

```
  public class CartItem implements Serializable {
    private int qty;
    private int productId;
    private String note;

    // getters/setters...
  }
```

Explanation

The resulting html code is rendered as

```
  <input type="hidden" name="cart.items[0].productId" value="1">
  <input type="text" name="cart.items[0].qty" value="2">
  <input type="text" name="cart.items[0].note" value="This is a fun book!">

  <input type="hidden" name="cart.items[1].productId" value="2">
  <input type="text" name="cart.items[1].qty" value="2">
  <input type="text" name="cart.items[1].note" value="You love this one">

  <input type="hidden" name="cart.items[2].productId" value="3">
  <input type="text" name="cart.items[2].qty" value="$item.qty">
  <input type="text" name="cart.items[2].note" value="">
```

Webwork will populate all the entries in Cart with the correct values.
In depth, the ParametersInterceptor would apply the form results to our model, leading to the call similar like

```
  ((CartItem) updateCartAction.getCart().getItems().get(0)).setProductId(1);
```

for the first shown line in the rendered result.

2. When the number of line items is unknown

For example, you want to allow the user to enter any number of ISBN, quanty and a note. You can replace ArrayList with XWorkList, which will automatically create new items if the index is greater than the size of the list.

3. Use Type Conversion

If you want more advanced way to do this, check out Type Conversion documentation.

Transparent web-app I18N

This page last changed on May 19, 2006 by scud.

As of WebWork 2.2, this interceptor is included with the normal distribution as the I18n Interceptor

Consider adding transparent i18 with simple on-the-fly locale switching to your appliction via I18NInterceptor.

The main idea:
Interceptor could track locale switch requests, persist selection in current session and set locale for all (or appropriate) actions invoked.

```
package neuro.util.xwork;

import com.opensymphony.xwork.ActionSupport;
import com.opensymphony.xwork.ActionInvocation;
import com.opensymphony.xwork.interceptor.Interceptor;

import java.util.Locale;

import org.apache.commons.logging.Log;
import org.apache.commons.logging.LogFactory;

/**
 * I18nInterceptor
 * @author Aleksei Gopachenko
 */
public class I18nInterceptor implements Interceptor
{
    protected static final Log log = LogFactory.getLog(I18nInterceptor.class);

    public static final String DEFAULT_SESSION_ATTRIBUTE = "WW_TRANS_I18N_LOCALE";
    public static final String DEFAULT_PARAMETER = "request_locale";

    protected String parameterName = DEFAULT_PARAMETER;
    protected String attributeName = DEFAULT_SESSION_ATTRIBUTE;

    /**
     */
    public I18nInterceptor()
    {
        if(log.isDebugEnabled()) log.debug("new I18nInterceptor()");
    }

    public void setParameterName(String parameterName) {
        this.parameterName = parameterName;
    }

    public void setAttributeName(String attributeName) {
        this.attributeName = attributeName;
    }

    /**
     */
    public void init() {
        if(log.isDebugEnabled()) log.debug("init()");
    }

    /**
     */
    public void destroy() {
        if(log.isDebugEnabled()) log.debug("destroy()");
    }

    /**
     */
    public String intercept(ActionInvocation invocation) throws Exception {
        if(log.isDebugEnabled()) log.debug("intercept '"
            +invocation.getProxy().getNamespace()+"/"
            +invocation.getProxy().getActionName()+"' { ");

        //get requested locale
        Object requested_locale =
invocation.getInvocationContext().getParameters().get(parameterName);
        if(requested_locale!=null && requested_locale.getClass().isArray() &&
((Object[])requested_locale).length==1) {
            requested_locale=((Object[])requested_locale)[0];
        }
        if(log.isDebugEnabled()) log.debug("requested_locale="+requested_locale);
        //save it in session
        if (requested_locale!=null) {
            Locale locale = (requested_locale instanceof Locale)?
              (Locale) requested_locale : localeFromString(requested_locale.toString());
            if(log.isDebugEnabled()) log.debug("store locale="+locale);
            if(locale!=null)invocation.getInvocationContext().getSession().put(attributeName,locale);
        }
        //set locale for action
        Object locale = invocation.getInvocationContext().getSession().get(attributeName);
        if (locale!=null && locale instanceof Locale) {
            if(log.isDebugEnabled()) log.debug("apply locale="+locale);
            invocation.getInvocationContext().setLocale((Locale)locale);
        }

        if(log.isDebugEnabled()) log.debug("before
Locale="+((ActionSupport)invocation.getAction()).getLocale());
        final String result = invocation.invoke();
        if(log.isDebugEnabled()) log.debug("after
Locale="+((ActionSupport)invocation.getAction()).getLocale());

        if(log.isDebugEnabled()) log.debug("intercept } ");
        return result;
    }

    Locale localeFromString(String localeStr) {
        if ((localeStr == null) || (localeStr.trim().length() == 0) || (localeStr.equals("_")))
{
            return Locale.getDefault();
        }
        int index = localeStr.indexOf('_');
        if (index < 0) {
            return new Locale(localeStr);
        }
        String language = localeStr.substring(0,index);
        if (index == localeStr.length()) {
            return new Locale(language);
        }
        localeStr = localeStr.substring(index +1);
        index = localeStr.indexOf('_');
        if (index < 0) {
            return new Locale(language,localeStr);
        }
        String country = localeStr.substring(0,index);
        if (index == localeStr.length()) {
            return new Locale(language,country);
        }
        localeStr = localeStr.substring(index +1);
        return new Locale(language,country,localeStr);
    }

}
```

Can be enabled for whole package via something like:

```
<interceptor name="i18n" class="neuro.util.xwork.I18nInterceptor">
    <!-- on which request parameter we should react, optional, defaults to "request_locale" ->
    <param name="parameterName">set_locale</param>
    <!- under which session attribute locale should be stored, optional, defaults to
"WW_TRANS_I18N_LOCALE" ->
    <param name="attributeName">ww_locale</param>
</interceptor>

<interceptor-stack name="i18nStack">
    <interceptor-ref name="i18n"/>
```

```
        <interceptor-ref name="defaultStack"/>
    </interceptor-stack>

    <default-interceptor-ref name="i18nStack"/>
```

...and invoked in web-app just by adding few links to menu:

```

    <a href="?set_locale=en">EN</a>
    <a href="?set_locale=ru">RU</a>
    <!- etc -->
```

Of course you still need to move all explisitly defined messages or labels to appropriate ResourceBundles and make translations. Be sure to check out your

- Actions - to use getText(...)
- *-validation.xml files - to use <message key=...>
- results/views - to use WW i18n services by <webwork:text ...> tag, or directly by evaluating getText(...) OGNL expression on current stack.

If this Interceptor is generally useful, may be it should go into codebase?

Using Checkboxes

This page last changed on May 19, 2006 by scud.

Using Checkboxes (General)

The biggest gotcha for newbies is that you must set the 'value' attribute in the html <input> field to use Checkboxes with WW. By default your browser will set this to some value. Firefox uses "on" - not sure what IE or others use. You must make this a sensible value for whatever property you are setting.

Using Checkboxes to set boolean fields

HTML:

```
<input type="checkbox" name="user.lockedOut" value="true"/>
```

If the user checks this box, the browser will send "user.lockedOut=true" in the QueryString and action.getUser().setLockedOut(true) will be called. If the user does not check the box, the browser will not send anything, so make sure that you have initialised lockedOut to false to start with.

```
private boolean m_lockedOut = false;

    public void setLockedOut(boolean lockedOut) { m_lockedOut = lockedOut; }
```

Using Checkboxes to set a collection

Our user has a number of priviliges that are stored as a Set of strings. To use checkboxes for these, we have HTML that looks like:

```
<input type="checkbox" name="user.priv" value="boss"/>
<input type="checkbox" name="user.priv" value="admin"/>
<input type="checkbox" name="user.priv" value="manager"/>
```

Say a user checks the first 2; the browser will send the query string: user.priv=boss&user.priv=admin.

OGNL will end up calling

```
action.getUser().setPriv(String[] {"boss", "admin"})
```

You can write this method like:

```
Set m_privileges = new HashSet();

    public void setPriv(String[] privs) {
        for (int i = 0; i < privs.length; i++) {
            m_privileges.add(privs[i]);
        }
    }
```

Full Detailed example:

This example uses a kind-of model-driven action (see Model Driven Interceptor). The action returns a single getter for the User object whose values are populated.

- Using Checkboxes \- EditAction.java
- Using Checkboxes \- Velocity and HTML
- Using Checkboxes \- User.java

Using Checkboxes - EditAction.java

This page last changed on Mar 30, 2006 by scud.

```
package cash.action;

import org.apache.log4j.Logger;

import cash.config.ConfigManager;
import cash.model.User;
import cash.util.HibernateUtil;
import cash.validator.PasswordFormatValidator;

import net.sf.hibernate.LockMode;

/**
 * Edit a user
 * @author Joel Hockey
 * @version $Id: $
 */
public class EditAction extends HibernateAction {
    private static final Logger LOG = Logger.getLogger(EditAction.class);

    private User m_user = new User();
    private String m_repeatPassword;

    /** return user to be edited. */
    public User getUser() { return m_user; }

    /** @param pwd repeat of password */
    public void setRepeatPassword(String pwd) { m_repeatPassword = pwd; }
    /** @return repeat password */
    public String getRepeatPassword() { return m_repeatPassword; }

    /** override super */
    public String execute() throws Exception {
        LOG.debug("EditAction started");

        // get original user from session, check that password is valid, update and save.
        User u = (User)get("user");
        HibernateUtil.currentSession().lock(u, LockMode.NONE);

        // check that password has actually changed before updating
        if (!PasswordFormatValidator.PASSWORD_MASK.equals(m_user.getPassword())) {
            if (!u.changePassword(m_user.getPassword())) {
                addFieldError("user.password", "password must be different to previous "
                    + ConfigManager.getConfig().getUser().getNoRepeatHistory() + " passwords");
                return INPUT;
            }
        }

        m_user.copy(u);
        HibernateUtil.currentSession().save(u);
        User loginUser = (User)get(LoginAction.LOGIN_USER);
        if (u.getId() == loginUser.getId()) {
            set(LoginAction.LOGIN_USER, u);
        }
        return SUCCESS;
    }
}
```

Using Checkboxes - User.java

This page last changed on Mar 30, 2006 by scud.

```
package cash.model;

import net.sf.hibernate.HibernateException;

import org.apache.log4j.Logger;

import java.security.GeneralSecurityException;
import java.security.MessageDigest;
import java.security.SecureRandom;
import java.util.ArrayList;
import java.util.Date;
import java.util.HashSet;
import java.util.Iterator;
import java.util.List;
import java.util.Locale;
import java.util.Set;
import java.util.SortedSet;
import java.util.TimeZone;
import java.util.TreeSet;

import cash.config.ConfigManager;
import cash.util.Hex;
import cash.util.HibernateUtil;
import cash.util.UtcDate;
import cash.validator.PasswordFormatValidator;

/**
 * Represents a User object.  Clients of this class should instantiate a User object with the
 * multi-arg constructor rather than using setters.
 *
 * @author Joel Hockey
 * @version $Id: $
 * @hibernate.class
 *      table="user"
 *      dynamic-update="true"
 *      optimistic-lock="version"
 */
public class User implements java.io.Serializable {
    private static final Logger LOG = Logger.getLogger(User.class);

    private static MessageDigest s_md5;
    private static SecureRandom s_random;

    private static final int MAX_LOGIN_FAILURE_COUNT = 20;
    private static final boolean RESET_LOCKED_OUT_AFTER_TIME = true;
    private static final long RESET_LOCKED_OUT_TIME = 1 * 60 * 60 * 1000; // 1 hour

    private int m_id;
    private int m_version;
    private String m_username;
    private String m_password;
    private Date m_passwordChangeDate;
    private String m_hashedPassword;
    private SortedSet m_passwordHistory = new TreeSet();
    private String m_salt;
    private byte[] m_saltBytes;
    private Date m_createDate;
    private String m_email;
    private Locale m_locale;
    private TimeZone m_timeZone;
    private String m_telephone;
    private Date m_lastSuccessfulLogin;
    private String m_lastSuccessfulLoginIp;
    private Date m_lastFailedLogin;
    private String m_lastFailedLoginIp;
    private int m_loginFailureCount;
    private int m_maxLoginFailureCount = MAX_LOGIN_FAILURE_COUNT;
    private boolean m_resetLockedOutAfterTime = RESET_LOCKED_OUT_AFTER_TIME;
    private long m_resetLockedOutTime = RESET_LOCKED_OUT_TIME;
    private boolean m_lockedOut = false;
    private boolean m_disabled = false;
    private boolean m_isSuperUser = false;
    private boolean m_passwordNeverExpires = false;
    private Set m_privileges = new HashSet();

    static {
        try {
            s_md5 = MessageDigest.getInstance("MD5");
            s_random = SecureRandom.getInstance("SHA1PRNG");
        } catch (GeneralSecurityException gse) {
            // shouldn't happen
            LOG.error("Error creating MD5 or SHA1PRNG", gse);
            throw new RuntimeException("Error creating MD5 or SHA1PRNG");
        }
    }

    /** default constructor for Hibernate */
    public User() { }

    /**
     * Create a User.
     *
     * @param username The username for logging in
     * @param password The user's password
     * @param email The user's email
     * @throws InvalidPasswordException if password is invalid.
     */
    public User(String username, String password, String email) throws InvalidPasswordException
{

        m_username = username;

        // password
        initSalt();
        if (!PasswordFormatValidator.checkPasswordFormat(password)) {
            throw new InvalidPasswordException();
        }
        m_hashedPassword = hashPassword(password);

        m_createDate = UtcDate.createUtcDate();
        m_email = email;
        m_locale = Locale.getDefault();
        m_timeZone = TimeZone.getDefault();
    }

    /** @param id The id to set */
    public void setId(int id) { m_id = id; }

    /**
     * @return unique id of this User.  Generated by DB.
     * @hibernate.id
     *      generator-class="native"
     */
    public int getId() { return m_id; }

    /** @param version The version of this object */
    public void setVersion(int version) { m_version = version; }

    /**
     * @return version of this object
     * @hibernate.version
     */
    public int getVersion() { return m_version; }

    /** @param username The username to set */
    public void setUsername(String username) { m_username = username; }

    /**
     * @return username
     * @hibernate.property
     *      length="32"
     *      unique="true"
     *      not-null="true"
     */
    public String getUsername() { return m_username; }

    /**
     * Set's the user's password without updating history or checking validity.
     * This should only be used at User creation time, and password validity
```

```
         * should be checked externally to this method.
         * Do not use to update password, see {@link #changePassword(String)}
         * @param password user's password
         */
        public void setPassword(String password) {
            m_password = password;
            if (m_salt == null) {
                initSalt();
            }
            m_hashedPassword = hashPassword(password);
            m_passwordChangeDate = UtcDate.createUtcDate();
        }

        /**
         * This method is provided to help at User creation time.  It will only return
         * valid values if {@link #setPassword(String)} has already been called.
         * @return plaintext password.
         */
        public String getPassword() { return m_password; }

        /** @param time Date (UTC) user last changed password. */
        public void setPasswordChangeDate(Date time) { m_passwordChangeDate = time; }

        /**
         * @return UTC date of last password change
         * @hibernate.property
         *      type="cash.model.TimestampType"
         *      length="23"
         */
        public Date getPasswordChangeDate() { return m_passwordChangeDate; }

        /**
         * Sets the user's hashed password.  This method is provided only for the use
         * of hibernate.  Users of this class should not call this method.
         * Use the {@link #setPassword(String)} method to set the plaintext password.
         * @param hash The hashed password to set
         */
        public void setHashedPassword(String hash) {
            m_hashedPassword = hash;
        }

        /**
         * @return hashed password
         * @hibernate.property
         *      column="pwd"
         *      length="32"
         *      not-null="true"
         */
        public String getHashedPassword() { return m_hashedPassword; }

        /**
         * @param oldPasswords The last n passwords, where n
         * is defined as noRepeatHistory in User configuration.  Passwords are ordered
         * in descending order of creation.
         */
        public void setPasswordHistory(SortedSet oldPasswords) { m_passwordHistory = oldPasswords;
    }

        /**
         * @return Password history
         * @hibernate.set
         *      lazy="true"
         *      sort="cash.model.PasswordHistory"
         *      inverse="true"
         *      cascade="all"
         * @hibernate.collection-key
         *      column="userId"
         * @hibernate.collection-one-to-many
         *      class="cash.model.PasswordHistory"
         */
        public SortedSet getPasswordHistory() { return m_passwordHistory; }

        /** @param random The random salt to be used with password */
        public void setSalt(String random) {
            m_salt = random;
            m_saltBytes = Hex.fromString(random);
        }
        /**
         * @return random salt used with password
         * @hibernate.property
         *      length="32"
         *      not-null="true"
         */
        public String getSalt() { return m_salt; }

        /** @param time create date */
        public void setCreateDate(Date time) { m_createDate = time; }

        /**
         * @return Date in UTC user was created.
         * @hibernate.property
         *      update="false"
         *      not-null="true"
         *      type="cash.model.TimestampType"
         *      length="23"
         */
        public Date getCreateDate() { return m_createDate; }

        /** @param email User's email */
        public void setEmail(String email) { m_email = email; }

        /**
         * @return User's email
         * @hibernate.property
         *      length="255"
         *      not-null="true"
         */
        public String getEmail() { return m_email; }

        /** @param locale The User's locale.  This should be a 2 character field. */
        public void setLocale(Locale locale) { m_locale = locale; }

        /**
         * @return User's locale.  Uses 2 character ISO-something value.
         * @hibernate.property
         *      not-null="true"
         */
        public Locale getLocale() { return m_locale; }

        /** @param timeZone User's time zone */
        public void setTimeZone(TimeZone timeZone) { m_timeZone = timeZone; }

        /**
         * @return User's timezone
         * @hibernate.property
         *      not-null="true"
         */
        public TimeZone getTimeZone() { return m_timeZone; }

        /** @param telephone User's telephone */
        public void setTelephone(String telephone) { m_telephone = telephone; }

        /**
         * @return Telephone of user
         * @hibernate.property
         *      length="16"
         */
        public String getTelephone() { return m_telephone; }

        /** @param time user's last successful login date in UTC. */
        public void setLastSuccessfulLogin(Date time) { m_lastSuccessfulLogin = time; }

        /**
         * @return UTC date of last successful login
         * @hibernate.property
         *      type="cash.model.TimestampType"
         *      length="23"
         */
        public Date getLastSuccessfulLogin() { return m_lastSuccessfulLogin; }

        /** @param ip IP address used for user's last successful login. */
        public void setLastSuccessfulLoginIp(String ip) { m_lastSuccessfulLoginIp = ip; }

        /**
         * @return IP address used for last successful login
```

```
     * @hibernate.property
     */
    public String getLastSuccessfulLoginIp() { return m_lastSuccessfulLoginIp; }

    /** @param time user's last failed login date in UTC. */
    public void setLastFailedLogin(Date time) { m_lastFailedLogin = time; }

    /**
     * @return UTC date of last failed login
     * @hibernate.property
     *      type="cash.model.TimestampType"
     *      length="23"
     */
    public Date getLastFailedLogin() { return m_lastFailedLogin; }

    /** @param ip IP address used for user's last failed login. */
    public void setLastFailedLoginIp(String ip) { m_lastFailedLoginIp = ip; }

    /**
     * @return IP address used for last failed login
     * @hibernate.property
     */
    public String getLastFailedLoginIp() { return m_lastFailedLoginIp; }

    /**
     * Sets the number of times that a user has failed when attempting to login.
     * This value is reset when a user logs in successfully, or their account is reset.
     * @param count the value to set.
     */
    public void setLoginFailureCount(int count) { m_loginFailureCount = count; }

    /**
     * @return The number of times that a user has failed when attempting to login.
     *  This value is reset when a user logs on successfully, or their account is reset.
     * @hibernate.property
     */
    public int getLoginFailureCount() { return m_loginFailureCount; }

    /**
     * @param count The maximum number of times that a user may fail to login before
     * their account is locked out
     */
    public void setMaxLoginFailureCount(int count) { m_maxLoginFailureCount = count; }

    /**
     * @return The maximum number of times that a user may fail to login before their account
     * is locked out.
     * @hibernate.property
     */
    public int getMaxLoginFailureCount() { return m_maxLoginFailureCount; }

    /**
     * @param reset Whether this user's account will be unlocked after a specified time when it
is locked
     * due to login failure.
     * @see #setResetLockedOutAfterTime(boolean) setResetLockedOutAfterTime
     */
    public void setResetLockedOutAfterTime(boolean reset) { m_resetLockedOutAfterTime = reset;
}

    /**
     * @return Whether this user's account will be unlocked after a specified time when it
     * is locked out due to login failure.
     * @see #getResetLockedOutAfterTime getResetLockedOutAfterTime
     * @hibernate.property
     */
    public boolean getResetLockedOutAfterTime() { return m_resetLockedOutAfterTime; }

    /**
     * @param time The time in millis between login attempts before login failure count is
reset.  Login failure
     * count will only be reset if the Reset Locked Out After Time boolean is set to true.
     */
    public void setResetLockedOutTime(long time) { m_resetLockedOutTime = time; }

    /**
     * @return Time in milliseconds before account is auto-reset after login lockout.
     * @hibernate.property
     */
    public long getResetLockedOutTime() { return m_resetLockedOutTime; }

    /** @param lockedOut User's locked out status. */
    public void setLockedOut(boolean lockedOut) { m_lockedOut = lockedOut; }

    /**
     * @return Whether this user's account is locked out
     * @hibernate.property
     */
    public boolean isLockedOut() { return m_lockedOut; }

    /** @param disabled User's disabled status. */
    public void setDisabled(boolean disabled) { m_disabled = disabled; }

    /**
     * @return Whether this user's account disabled
     * @hibernate.property
     */
    public boolean isDisabled() { return m_disabled; }

    /** @param superUser True if user is super user */
    public void setSuperUser(boolean superUser) { m_isSuperUser = superUser; }

    /**
     * @return Whether this user is a super user
     * @hibernate.property
     */
    public boolean isSuperUser() { return m_isSuperUser; }

    /** @param expires True if user's password never expires */
    public void setPasswordNeverExpires(boolean expires) { m_passwordNeverExpires = expires; }

    /**
     * @return Whether this user's password ever expires
     * @hibernate.property
     */
    public boolean getPasswordNeverExpires() { return m_passwordNeverExpires; }

    /** @param privs Set of privileges for this user  */
    public void setPrivileges(Set privs) { m_privileges = privs; }

    /**
     * @return Set of Privileges for this User.
     * @hibernate.set
     *      table="user_priv"
     *      lazy="true"
     *      cascade="all"
     * @hibernate.collection-key
     *      column="userId"
     * @hibernate.collection-element
     *      column="priv"
     *      type="string"
     */
    public Set getPrivileges() { return m_privileges; }

    /** convenience method of OGNL */
    public void setPriv(String[] privs) {
        for (int i = 0; i < privs.length; i++) {
            m_privileges.add(privs[i]);
        }
    }


// other methods

    /**
     * Changes the user's password.  Password must meet criteria
     * defined in configuration.  The user's password will be appended to
     * a random 20 byte salt and then hashed using MD5 to create the
     * value that will be stored in the DB.  The current Hibernate Session
     * will be used to update pwd history.
     *
     * @param password The password to set
     * @return true if password is changed, false if password was not changed
     * because it did not meet password requirements.
     * @throws HibernateException if error updating password history
     */
    public boolean changePassword(String password) throws HibernateException {
        // check format
        if (!PasswordFormatValidator.checkPasswordFormat(password)) {
            return false;
        }

        // check history
        // first check current password
        String hashedPwd = hashPassword(password);
        LOG.debug("checking if password is same as current");
        if (hashedPwd.equals(m_hashedPassword)) {
            LOG.info("password is same as current password");
            return false;
        }

        LOG.debug("checking if password exists in history.  History size is " +
m_passwordHistory.size());
        for (Iterator i = getPasswordHistory().iterator(); i.hasNext(); ) {
            PasswordHistory ph = (PasswordHistory)i.next();
            if (hashedPwd.equals(ph.getHashedPassword())) {
                LOG.info("password already used as one of last "
                    + ConfigManager.getConfig().getUser().getNoRepeatHistory());
                return false;
            }
        }

        // add current pwd to history and truncate history if it is too long now
        PasswordHistory ph = new PasswordHistory(this, m_hashedPassword);
        m_passwordHistory.add(ph);
        LOG.debug("saving old password into password history");
        HibernateUtil.currentSession().save(ph);
        // compare to (noRepeat - 1) because we are checking current as part of history
        if (m_passwordHistory.size() > ConfigManager.getConfig().getUser().getNoRepeatHistory()
- 1) {
            PasswordHistory toRemove = (PasswordHistory)m_passwordHistory.first();
            LOG.info("Removing password history object for user " + m_username
                    + " created: " + toRemove.getCreateDate());
            m_passwordHistory.remove(toRemove);
            HibernateUtil.currentSession().delete(toRemove);
        }

        // now set password and date
        m_hashedPassword = hashedPwd;
        m_passwordChangeDate = UtcDate.createUtcDate();
        return true;
    }

    /**
     * Hashes input pwd to see if it equals stored pwd hash value.
     * @param pwd Password to check
     * @return true if passwords are equal.
     */

    public boolean passwordEquals(String pwd) {
        String hash = hashPassword(pwd);
        return m_hashedPassword.equalsIgnoreCase(hash);
    }

    /**
     * Hashes salt and password to produce hashed password.
     * @param pwd Password to hash
     * @return Hex encoding of MD5 hash of salt and pwd
     */
    private String hashPassword(String pwd) {
        byte[] pwdBytes = pwd.getBytes();  //TODO:  should an encoding be specified here?
        byte[] in = new byte[OS:m_saltBytes.length + pwdBytes.length];
        System.arraycopy(m_saltBytes, 0, in, 0, m_saltBytes.length);
        System.arraycopy(pwdBytes, 0, in, m_saltBytes.length, pwdBytes.length);
        byte[] out = s_md5.digest(in);
        return Hex.toString(out);
    }

    /** initialises salt */
    private void initSalt() {
        m_saltBytes = new byte[OS:16];
        s_random.nextBytes(m_saltBytes);
        m_salt = Hex.toString(m_saltBytes);
    }
    /** @return String representation of User */
    public String toString() {
        StringBuffer sb = new StringBuffer(500);
        sb.append("[").append("ID:").append(m_id)
        .append(",version:").append(m_version)
        .append(",hashedPassword:").append(m_hashedPassword)
        .append(",salt:").append(m_salt)
        .append(",createDate:").append(m_createDate)
        .append(",email:").append(m_email)
        .append(",locale:").append(m_locale)
        .append(",timeZone:").append(m_timeZone)
        .append(",telephone:").append(m_telephone)
        .append(",lastSuccessfulLogin:").append(m_lastSuccessfulLogin)
        .append(",lastSuccessfulLoginIp:").append(m_lastSuccessfulLoginIp)
        .append(",lastFailedLogin:").append(m_lastFailedLogin)
        .append(",lastFailedLoginIp:").append(m_lastFailedLoginIp)
        .append(",loginFailureCount:").append(m_loginFailureCount)
        .append(",maxLoginFailureCount:").append(m_maxLoginFailureCount)
        .append(",resetLockedOutAfterTime:").append(m_resetLockedOutAfterTime)
        .append(",resetLockedOutTime:").append(m_resetLockedOutTime)
        .append(",lockedOut:").append(m_lockedOut)
        .append(",disabled:").append(m_disabled)
        .append(",isSuperUser:").append(m_isSuperUser)
        .append(",passwordNeverExpires:").append(m_passwordNeverExpires)
        .append(",passwordChangeDate:").append(m_passwordChangeDate)
        .append(",privs:").append(m_privileges);
        return sb.toString();
    }

    /**
     * Copies editable data from this object to User object provided.  This is used
     * in Edit actions.  Not all fields are copied, only those that are editable
     * @param user Object to copy to
     */
    public void copy(User user) {
        user.setUsername(m_username);
        user.setEmail(m_email);
        user.setLocale(m_locale);
        user.setTimeZone(m_timeZone);
        user.setTelephone(m_telephone);
        user.setLockedOut(m_lockedOut);
        user.setDisabled(m_disabled);
        user.setPasswordNeverExpires(m_passwordNeverExpires);

        // do some smarts for privs removal.  Clear all if more than half are removed
        if (m_privileges.size() <= user.getPrivileges().size() / 2) {
            LOG.debug("detected that many privs are removed, clearing all");
            user.setPrivileges(m_privileges);
        } else {
            // find which ones should be removed
            List toRemove = new ArrayList();
            for (Iterator i = user.getPrivileges().iterator(); i.hasNext(); ) {
                String priv = (String)i.next();
                if (!m_privileges.contains(priv)) {
                    toRemove.add(priv);
                }
            }

            // remove them
            for (int i = 0; i < toRemove.size(); i++) {
                user.getPrivileges().remove(toRemove.get(i));
            }

            // add all new privs
            for (Iterator i = m_privileges.iterator(); i.hasNext(); ) {
                user.getPrivileges().add(i.next());
            }
        }
    }
}
```

Using Checkboxes - Velocity and HTML

This page last changed on Mar 30, 2006 by scud.

Velocity View - edit.vm:

```
<html>
<body onload="document.forms[0].elements[0].focus()">

<a href="home.vm">Home</a><br/>

#if ($fieldErrors)
  #foreach ($error in $fieldErrors)
    $error<br>
  #end
#end
#if ($actionErrors)
  #foreach ($error in $actionErrors)
    $error<br>
  #end
#end

<form name="edit" action="edit.action" method="post">
<table>
<tr><td>Name</td><td>$user.username</td></tr>
#formRowText("Password" "user.password"
$stack.findValue("@cash.validator.PasswordFormatValidator@PASSWORD_MASK"))
#formRowText("Repeat Password" "repeatPassword"
$stack.findValue("@cash.validator.PasswordFormatValidator@PASSWORD_MASK"))
#formRowText("Email" "user.email" $!user.email)
#formRowSelect("Language" "user.locale"
$stack.findValue("@cash.util.Html@getInstance()").getLocales($locale) $!user.locale.toString())
#formRowSelect("Time Zone" "user.timeZone"
$stack.findValue("@cash.util.Html@getInstance()").getTimeZones($locale) $!user.timeZone.ID)
#formRowText("Telephone" "user.telephone" $!user.telephone)
#formRowCheckbox("Locked Out" "user.lockedOut" "true" $user.lockedOut)
#formRowCheckbox("Disabled" "user.disabled" "true" $user.disabled)


#set ($privs = [OS:"boss", "admin", "early", "late", "train"])

#foreach ($priv in $privs)
  #set ($checked = $user.privileges.contains($priv))
  #formRowCheckbox($priv "user.priv" $priv $checked)
#end
<tr><td>&nbsp;</td><td><input type="submit" name="submit" value="submit"></td></tr>
</table>

<input type="hidden" name="user.username" value="$user.username">
</form>

</body>
</html>
```

Velocity Macros - macros.vm:

```
#macro (formRowText $label $name $value)
  <tr><td><label for="$name">$label</label></td><td><input id="$name" type="text" name="$name"
value="$!value"></td></tr>
#end

#macro (formRowSelect $label $name $options $selectedValue)
  <tr><td><label for="$name">$label</label></td><td><select id="$name" name="$name">
#foreach ($option in $options)
<option#if ($option.get(0).equals($selectedValue)) selected#end
value="$option.get(0)">$option.get(1)</option>
#end
</select></td></tr>
#end
```

```
#macro (formRowCheckbox $label $name $value $checked)
  <tr><td><label for="$name.$value">$label</label></td><td><input id="$name.$value"
type="checkbox" name="$name" value="$value"#if ($checked) checked#end ></td></tr>
#end
```

Note that I don't use the webwork UI tags. (The HTML that comes out of them looks like vomit.)

The HTML generated from above looks like:

```
<html>
<body onload="document.forms[0].elements[0].focus()">

<a href="home.vm">Home</a><br/>


<form name="edit" action="edit.action" method="post">
<table>
<tr><td>Name</td><td>user</td></tr>
  <tr><td><label for="user.password">Password</label></td><td><input id="user.password"
type="text" name="user.password" value="********"></td></tr>
  <tr><td><label for="repeatPassword">Repeat Password</label></td><td><input
id="repeatPassword" type="text" name="repeatPassword" value="********"></td></tr>

  <tr><td><label for="user.email">Email</label></td><td><input id="user.email" type="text"
name="user.email" value="user@example.com"></td></tr>
  <tr><td><label for="user.locale">Language</label></td><td><select id="user.locale"
name="user.locale">
<option value="en">English</option>
<option selected value="en_AU">English (Australia)</option>
<option value="en_US">English (United States)</option>
<option value="en_GB">English (United Kingdom)</option>
<option value="es">Spanish</option>
<option value="fr">French</option>

<option value="de">German</option>
</select></td></tr>
  <tr><td><label for="user.timeZone">Time Zone</label></td><td><select id="user.timeZone"
name="user.timeZone">
<option selected value="America/Los_Angeles">(GMT-08:00) Los Angeles</option>
<option value="Europe/London">(GMT+00:00) London</option>
<option value="Australia/Brisbane">(GMT+10:00) Brisbane</option>
</select></td></tr>
  <tr><td><label for="user.telephone">Telephone</label></td><td><input id="user.telephone"
type="text" name="user.telephone" value="134"></td></tr>
  <tr><td><label for="user.lockedOut.true">Locked Out</label></td><td><input
id="user.lockedOut.true" type="checkbox" name="user.lockedOut" value="true" ></td></tr>

  <tr><td><label for="user.disabled.true">Disabled</label></td><td><input
id="user.disabled.true" type="checkbox" name="user.disabled" value="true" ></td></tr>



    <tr><td><label for="user.priv.boss">boss</label></td><td><input id="user.priv.boss"
type="checkbox" name="user.priv" value="boss" ></td></tr>
    <tr><td><label for="user.priv.admin">admin</label></td><td><input id="user.priv.admin"
type="checkbox" name="user.priv" value="admin" ></td></tr>
    <tr><td><label for="user.priv.early">early</label></td><td><input id="user.priv.early"
type="checkbox" name="user.priv" value="early" ></td></tr>
    <tr><td><label for="user.priv.late">late</label></td><td><input id="user.priv.late"
type="checkbox" name="user.priv" value="late" ></td></tr>

    <tr><td><label for="user.priv.train">train</label></td><td><input id="user.priv.train"
type="checkbox" name="user.priv" value="train" ></td></tr>
<tr><td>&nbsp;</td><td><input type="submit" name="submit" value="submit"></td></tr>
</table>

<input type="hidden" name="user.username" value="user">
</form>

</body>
</html>
```

Using Maven to set up an Eclipse project for Webwork

This page last changed on Mar 30, 2006 by scud.

Because WebWork is under active development, these instructions are likely to be out-of-date in specifics. Hopefully the basic strategies will still apply.

First, if xwork is not available from a Maven repo (for example, if it has moved to a SNAPSHOT dependency), then check it out from the CVS repository and run "mvn install" - this gives you the snapshot of XWork as well as the POM for resolving transitive dependencies like oscore. This may apply to other dependencies which are not on ibiblio, but when I first tried this, xwork was the one with enough transitive dependencies to be hard to manage any other way.

For other dependencies which are not on ibiblio, from the webwork CVS checkout (sandbox), run "ant common.jar". This will cause Ivy to fetch the other dependencies you need.

Then, for each of the missing dependencies, install using "mvn install:install-file" Below are examples, but of course the path to your ".ivy-cache" directory will differ, and the versions are likely to change for lots of reason, but especially if the dependency is a SNAPSHOT with a timestamp in the filename.

```
mvn install:install-file -DartifactId=dwr -DgroupId=dwr -Dpackaging=jar -Dversion=1.1.3-beta \
    -Dfile=/Users/germuska/.ivy-cache/dwr/dwr/jars/dwr-1.1-beta-3.jar

mvn install:install-file -DartifactId=plexus-container-default -DgroupId=org.codehaus.plexus \
    -Dpackaging=jar -Dversion=1.0-alpha-10-SNAPSHOT \
-Dfile=/Users/germuska/.ivy-cache/org.codehaus.plexus/plexus-container-default/jars/plexus-container-default
```

If you, like me, are not running Java 5.0, then you will also need to install the dom3 APIs. As far as I can tell, the compiled JAR is not on any Maven repository. the source for dom3 can be found here: http://ibiblio.org/maven2/xerces/dom3-xml-apis/1.0/dom3-xml-apis-1.0-sources.jar You can build a jar from it and then put it in your own repository. After that, I had to edit webwork's pom.xml to point to the dependency -- webwork has a comment in the <profiles> section acknowledging the need for this, but presumably it will not be changed until dom3 gets officially loaded as a compiled JAR to some maven repository. I believe there should be a way to use Maven2's settings.xml file to do this locally without editing pom.xml, but I have not had a chance to investigate this yet.

These steps worked for me a week or so ago; the pom has changed since then and I haven't yet had time to rebuild, but this kind of approach should work.

Using WebWork and XWork with JSP 2.0 and JSTL 1.1

This page last changed on Mar 30, 2006 by scud.

WW2/WX1 and its taglib is oriented towards OGNL, which is using a value stack
for all action properties. These values are not direct available for the
expression language of JSP2/JSTL1.1.

However, it's easy to populate the request
attribute set, with all gettable properties of an action object. You need to provide
an interceptor that does the job, by register a PreResultListener which is
invoked after the return of Action.execute() but before the rendering of the result .

The interceptor below is using Jakarta BeanUtils. It first extracts all getters
of the current action, invokes them one at the time and stores the values into a map.
Then it iterates over the map and populates the request attribute set.
The double iteration is not needed, it's just there for clarity.

class ActionPropertyExportInterceptor

```
package com.whatever.interceptors;

import com.opensymphony.webwork.WebWorkStatics;
import com.opensymphony.xwork.Action;
import com.opensymphony.xwork.ActionInvocation;
import com.opensymphony.xwork.interceptor.AroundInterceptor;
import com.opensymphony.xwork.interceptor.PreResultListener;
import org.apache.commons.beanutils.PropertyUtils;
import javax.servlet.http.HttpServletRequest;
import java.beans.PropertyDescriptor;
import java.util.*;

/**
 * Populates HTTP Request Attributes with all gettable properties of the current action.
 */
public class ActionPropertyExportInterceptor extends AroundInterceptor {
    protected void before(ActionInvocation invocation) throws Exception {
        invocation.addPreResultListener( new PropertyExporter() );
    }
    protected void after(ActionInvocation dispatcher, String result) throws Exception { }

    public static class PropertyExporter implements PreResultListener {
        private static final List   ignore = Arrays.asList(new String[] {"class", "texts"});
//skip getClass,...

        //Invoked after Action.execute() but before Result
        //Calls all getters of the action and insert the values into the request
        public void beforeResult(ActionInvocation invocation, String resultCode) {
            Map                 props   = extractGetterPropertyValues( invocation.getAction()
);
            HttpServletRequest  request = getRequest(invocation);
            for (Iterator it = props.entrySet().iterator(); it.hasNext();) {
                Map.Entry   e = (Map.Entry) it.next();
                request.setAttribute((String) e.getKey(), e.getValue());
            }
        }

        public Map extractGetterPropertyValues(Object bean) {
            PropertyDescriptor[]  descr = PropertyUtils.getPropertyDescriptors(bean);
            Map                   props = new HashMap();
            for (int i = 0; i < descr.length; i++) {
                PropertyDescriptor d = descr[i];
                if (d.getReadMethod() == null) continue;
                if (ignore.contains(d.getName())) continue;

                try {
                    props.put(d.getName(), PropertyUtils.getProperty(bean, d.getName()));
                } catch (Exception e) { }
            }
            return props;
        }

        public HttpServletRequest getRequest(ActionInvocation invocation) {
            return (HttpServletRequest)
invocation.getInvocationContext().get(WebWorkStatics.HTTP_REQUEST);
        }
    }
}
```

Don't forget to declare the interceptor in your xwork.xml file and insert it
into your interceptor stack.

xwork.xml snippet

```
<interceptor name="export" class="com.whatever.interceptors.ActionPropertyExportInterceptor" />
. . .
<interceptor-stack name="standard-interceptors">
    <interceptor-ref name="timer" />
    <interceptor-ref name="logger" />
    <interceptor-ref name="params" />
*    <interceptor-ref name="export"/>*
    <interceptor-ref name="validateParams"/>
    <interceptor-ref name="awarePlugger" />
</interceptor-stack>
```

Your action need to provide getters for all properties that should be exported into the
request attribute set.

class ViewUser

```
public class ViewUser extends ActionSupport {
    private int                         id;
    private User                        user;

    public String execute() throws Exception {
        user = findUser( getId() );
        return Action.SUCCESS;
    }

    public  int   getId()          {return id;}
    public  void  setId(int id)    {this.id = id;}
*    public  User  getUser()        {return user;}*

    private User  findUser(int id) {...}
}
```

The User class might look like this

class User

```
import java.util.Date;
public class User {
    private int     id;
    private String  firstName, lastName, email;
    private String  street, zip, city;
    private Date    date;

    public String  getFirstName() {return firstName;}
    //..._getters and setters_...
}
```

Finally, using the samples above you can write your JSP2 page like this.

ViewUser.jsp

```
<%@ taglib prefix="c"   uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<%@ taglib prefix="fn"  uri="http://java.sun.com/jsp/jstl/functions" %>
<html>
<head>
    <title>Info about ${user.firstName}</title>
</head>
<body>
    <h1>Info about ${user.firstName} ${user.lastName} [OS:ID=${user.id}]</h1>
    <table border="1" cellspacing="0" cellpadding="2" width="90%" >
    <tr>
        <th>Name</th> <td>${user.firstName} ${user.lastName}</td>
    </tr>
    <tr>
        <th>Created</th> <td><fmt:formatDate value="${user.date}" pattern="yyyy-MM-dd
HH:mm"/></td>
    </tr>
    <tr>
        <th>Email</th> <td>${user.email}</td>
    </tr>
    <tr>
        <th>Address</th> <td>${user.street} ${user.zip} ${fn:toUpperCase(user.city)}</td>
    </tr>
    </table>
</body>
</html>
```

Displaying validation errors with JSTL

```
<c:if test="${!empty fieldErrors || !empty actionErrors}">
  <div class="red">
    <ul>
      <c:forEach items="${fieldErrors}" var="fieldError">
        <c:forEach items="${fieldError.value}" var="error">
          <li>${error}</li>
        </c:forEach>
      </c:forEach>
      <c:forEach items="${actionErrors}" var="actionError">
        <li>${actionError}</li>
      </c:forEach>
    </ul>
  </div>
</c:if>
```

Using WebWork Components

This page last changed on Mar 30, 2006 by scud.

A simple example of using WebWork components is available in the webwork-example.war that comes with the WebWork 2.0 Beta 1 distribution. You can download the distribution from https://webwork.dev.java.net/servlets/ProjectDocumentList .

Components are defined _/WEB-INF/classes/components.xml_.

The example consists of one component, which is defined by

```
<component>
  <scope>session</scope>
  <class>com.opensymphony.webwork.example.counter.Counter</class>
  <enabler>com.opensymphony.webwork.example.counter.CounterAware</enabler>
</component>
```

com.opensymphony.webwork.example.counter.Counter is just a POJO.

com.opensymphony.webwork.example.counter.CounterAware is an interface which your *Action classes have to implement.

```
public interface CounterAware {
    public void setCounter(Counter counter);
}
```

Additionally, you need to tag your actions with the intercepter, for example,

```
<action name="SimpleCounter" class="com.opensymphony.webwork.example.counter.SimpleCounter">
  <result name="success" type="dispatcher">
    <param name="location">/success.jsp</param>
  </result>
  <interceptor-ref name="defaultComponentStack"/>
</action>
```

WebWork will call the interface and set the Counter bean . The Counter bean would then be subsequently be available to be used by your *Action classes.

Value Stack Internals

This page last changed on Mar 30, 2006 by scud.

As Matt Ho explained on the mailing list:

A value stack is essentially a List. Calling [1] on the stack,
returns a substack beginning with the element at index 1. It's only
when you call methods on the stack that your actual objects will be
called.

Said another way, let's say I have a value stack that consists of a
model and an action as follows:

[model, action]

here's how the following ognl would resolve:

[0] - a CompoundRoot object that contains our stack, [model, action]

[1] - another CompoundRoot that contains only [action]

[0].toString() - calls toString() on the first object in the value
stack (excluding the CompoundRoot) that supports the toString() method

[1].foo - call getFoo() on the first object in the value stack
starting from [OS:action] and excluding the CompoundRoot that supports a
getFoo() method

I hope this doesn't sound too confusing :\

If you're using Velocity, this can most easily be written as:

$stack.findValue("[0]").peek()

Unfortunately, <ww:property value="[0].peek()"/> won't work as this
would translate into "starting at the top of the value stack (and
excluding the CompoundRoot), find the first object that has a method
called peek()"

--------thanks Matt!

here is the com.opensymphony.xwork.util.CompoundRoot class which Matt mentions:

```
public class CompoundRoot extends ArrayList {
    //~ Constructors ///////////////////////////////////////////////////////////

    public CompoundRoot() {
    }

    public CompoundRoot(List list) {
        super(list);
    }
    //~ Methods ////////////////////////////////////////////////////////////////

    public CompoundRoot cutStack(int index) {
        return new CompoundRoot(subList(index, size()));
    }

    public Object peek() {
        return get(0);
    }

    public Object pop() {
        return remove(0);
    }

    public void push(Object o) {
        add(0, o);
    }
}
```

What's on the stack?

NOTE: When rendering Freemarker / Velocity templates or result,
WebWork2 contains the following items by default in the ValueStack:

- req - the current HttpServletRequest
- res - the current HttpServletResponse
- stack - the current OgnlValueStack
- ognl - an instance of OgnlTool
- ui - a (now deprecated) instance of a ui tag renderer

@See com.opensymphony.webwork.views.util.ContextUtil

Webwork 2 HTML form buttons Howto

This page last changed on Mar 30, 2006 by scud.

HTML Form Buttons and Webwork2

This Howto will describe the usage of HTML form buttons to invoke different behavior in actions.

Determine which button was pressed

The trick is that the type conversion of XWork can be used to test which button was pressed in a simple way. When a button is pressed, a parameter is set in webwork with the name and value that are specified as the name and value attributes of your HTML button. XWork converts this automatically to boolean value if an appropriate property of the Action is found.
These boolean Properties can be tested to determine which button was pressed:

```
<form action="MyAction.action">
<input type="submit" name="buttonOnePressed" value="First option">
<input type="submit" name="buttonTwoPressed" value="Alternative Option">
</form>
```

```
public class MyAction extends Action {

    /**
     * Action implementation
     *
     * Sets the message according to which button was pressed.
     **/
    public String execute() {
        if (buttonOnePressed) {
            message="You pressed the first button";
        } else if (buttonTwoPressed) {
            message="You pressed the second button";
        } else {
            return ERROR;
        }
        return SUCCES;
    }

    // Input parameters
    private boolean buttonOnePressed=false;
    private boolean buttonTwoPressed=false;

    public void setButtonOnePressed(boolean value) {
        this.buttonOnePressed = value;
    }


    public void setButtonTwoPressed(boolean value) {
        this.buttonTwoPressed = value;
    }

    // Output parameters

    private String message;
    public String getMessage() {
        return message;
    }
}
```

*Note_: Do not use String properties with buttons and test for the value that's set. This will break as soon as the _value attribute of the HTML button changes! This is likely because the value attribute used as the text shown to the user.

Dynamic set of buttons

Consider a web page showing a shopping cart or similiar tabular data. Often there is a button belonging to each row, in case of the shopping cart a delete button to remove the item from the cart. The number of buttons is dynamic and the id that couples the button to an item cannot go to the value attribute because all buttons should read "delete".

The solution is to __name* the buttons like delete[123], delete[594], delete[494] where 123, 594 and 494 are e.g. the items' ids:

```
<form action="UpdateCart.action">
  <ww:iterate value="items">
    <ww:property value="name">
    <input type="submit" name="delete[<ww:property value='id'>]" value="delete" /> <br/>
  </ww:iterate>
</form>
```

When e.g. the button for the item with the property id == "27" is pressed, a parameter named delete[27] and value "delete" is set in your action. The trick is to us declare your action's property "delete" as java.util.Map. Then, a key will exist for the button that was pressed:

```
public void class UpdateCart implements Action {
    // must be initialized to be usable as a webwork input parameter
    private Map delete = new HashMap();

    /** This is somewhat counter intuitive. But a property like "delete[OS:27]"
     *  that is set to "delete" by webwork will be interpreted by the underlying
     *  OGNL expression engine as "set the property 27 of the action's property
     *  "delete" to the value "delete". So we must provide a getter for this
     */ action. A setter is not needed.
    public Map getDelete() {
        return delete;
    }

    public String execute() {
        for(Iterator i = delete.keySet().iterator(); i.hasNext(); ) {
            String id = (String) i.next();
            ...
            // do what ever you want
            ...
         }
         ...
    }
}
```

In this case it would not be necessary to iterate the whole keySet because it contains only one key but the same code can be use to handle sets of checkboxes if this is prefered later:

```
<form action="UpdateCart.action">
  <ww:iterate value="items">
    <ww:property value="name">
    <input type="checkbox" name="delete[<ww:property value='item'/>]" value="delete"> <br/>
  </ww:iterate>
  <input type="submit" name="updateCart" value="Update the cart"/>
</form>
```

The two implementations can even be combined two provide a quick "delete this item" button and a set of checkboxes for "mass updates". All with the above code, cool eh?

Webwork 2 skinning

This page last changed on Mar 30, 2006 by scud.

Skinning in Webwork 2 can be done more than one way. We will show how to use two skins called "html" and "wml", and we'll be working with the following directory structure:

```
/WEB-INF
   /web.xml
/html
   /index.jsp
   /Register.jsp
/wml
   /index.jsp
   /Register.jsp
/index.jsp
```

Classic Approach

If you want to go the Webwork 1.3 route, simply place all actions in the default namespace so that they are accessible from any URL path. When you create your views, place them in the sub-directory that corresponds with the skin's identifier.

Your action configuration would look like this (simplified, without defined interceptors):

```
<package name="default">
   <action name="registration" class="x.actionset.Register">
      <result name="success" type="dispatcher">
         <param name="location">Register.jsp</param>
      </result>
      <interceptor-ref name="defaultStack"/>
   </action>
</package>
```

If a user requested http://yoursite/html/register.action, he would see the JSP located at /html/Register.jsp.

Namespace Defined

If you require the use of namespaces, you can do the following:

Simplified configuration example:

```
<package name="user" extends="default">
   <action name="register" class="x.x.actionset.Register">
      <result name="success" type="dispatcher">
         <param name="location">Register.jsp</param>
      </result>

      <interceptor-ref name="defaultStack"/>
  </action>
</package>

<package name="user-html" extends="user" namespace="/user/html" />
<package name="user-wml" extends="user" namespace="/user/wml" />
```

The last two package definitions extend the first package, changing only the namespace. The view result defined in the "register" action has a relative path. Because of this, you'll get the same behavior as the Classic Approach, but with the security of knowing that ONLY those two paths can be accessed for the action, instead of ANY path.

Webwork file upload handling

This page last changed on Mar 30, 2006 by scud.

File upload using WebWork

Webwork comes with built in file upload support. Uploading a file is simple. When ServletDispatcher begins it checks to see if the request contains multipart content. If it does the dispatcher creates a MultipartWrapperRequest. This wrapper handles receiving the file and saving to disk. It is important for the action programmer to check to see if any errors occured during processing. Three properties can be set that effect file uploading.

Properties

Webwork properties can be set by putting a file 'webwork.properties' in WEB-INF/classes. Any property found there will override the default value.

1. webwork.multipart.parser - This should be set to a class that extends MultiPartRequest. Currently WebWork ships with two implementations. "com.opensymphony.webwork.dispatcher.multipart.PellMultiPartRequest" and "com.opensymphony.webwork.dispatcher.multipart.CosMultiPartRequest" If the property is not found the Pell parser is used.
2. webwork.multipart.saveDir - The directory where the uploaded files will be placed. If this property is not set it defaults to javax.servlet.context.tempdir.
3. webwork.multipart.maxSize - The maximum file size in bytes to allow for upload. This helps prevent system abuse by someone uploading lots of large files. The default value is 2 Megabytes and can be set as high as 2 Gigabytes (higher if you want to edit the Pell multipart source but you really need to rethink things if you need to upload files larger then 2 Gigabytes!) If you are uploading more than one file on a form the maxSize applies to the combined total, not the individual file sizes.

If you're happy with the defaults there is no need to put any of the properties in webwork.properties. Here is my current webwork.properties

```
# don't really need to set this but I put it here for testing
# various values
webwork.multipart.parser=com.opensymphony.webwork.dispatcher.multipart.PellMultiPartRequest

# put the uploaded files in /tmp. My application will move them to their
# final destination
webwork.multipart.saveDir=/tmp
```

Note, while you can set these properties to new values at runtime the MultiPartRequestWrapper is created and the file handled before your action code is called. So if you want to change values you must do so before this action.

Sample form

```
<%@ taglib uri="webwork" prefix="ww" %>

<html>
  <head>
   <title>File Upload Test</title>
  </head>
  <body>
    <h1>File Upload</h1>
    <form action="FileUpload.action" method="POST" enctype="multipart/form-data">

    <center>
      <table width="350" border="0" cellpadding="3" cellspacing="0">
      <tr>
        <td colspan="2"><input type="file" name="FileName" value="Browse..." size="50"/></td>
      </tr>
      <tr>
        <td colspan="2" align="center">
          <input type="submit" value="Submit">
        </td>
      </tr>
      </table>
    </center>
  </form>

</body>
</html>
```

That's all you have to do to upload a file. No coding required, the file will be placed in the default directory. However, that leaves us with no error checking among other things. So let's add some code to the Action.

FileUploadAction.java

Before the action method is called the dispatcher will upload the file. Then we can get access to information about the file from MultiPartRequestWrapper.

```
MultiPartRequestWrapper multiWrapper =
                (MultiPartRequestWrapper) ServletActionContext.getRequest();
```

The first thing you should always do is check for errors. If there were any there's no point in continuing, most methods will return null. Unfortunately, currently there is no easy way to distinguish what error occured making it more difficult to route to different error pages. (I have improving error handling for file uploads on my stack of things I'd like to do sometime).

```
if (multiWrapper.hasErrors()) {
  Collection errors = multiWrapper.getErrors();
  Iterator i = errors.iterator();
  while (i.hasNext()) {
    addActionError((String) i.next());
  }
  return ERROR;
}
```

Now get the input tag name for the uploaded file and use that to get information on the transfer. Since you can upload multiple files (just add multiple input tags) at a time getFileNames returns an Enumeration of the names.

```
Enumeration e = multiWrapper.getFileNames();

while (e.hasMoreElements()) {
   // get the value of this input tag
   String inputValue = (String) e.nextElement();

   // get the content type
   String contentType = multiWrapper.getContentType(inputValue);

   // get the name of the file from the input tag
   String fileName = multiWrapper.getFilesystemName(inputValue);
```

```
      // Get a File object for the uploaded File
      File file = multiWrapper.getFile(inputValue);

      // If it's null the upload failed
      if (file == null) {
         addActionError("Error uploading: " + multiWrapper.getFilesystemName(inputValue));
      }

      // Do additional processing/logging...
   }
```

Further improvements.

Code above may be packed into one nice reusable component (Interceptor) that handles 90% of all typical file upload tasks. And Action does not know anything about web-app and just gets its files. Neat. See WW:File Upload Interceptor

Dependencies

This page last changed on Mar 30, 2006 by scud.

General information on dependencies

As with most modern and robust frameworks, WebWork has a number of external dependencies. However, in the case of WebWork, only a fraction of them are required. You can determine exactly what these dependencies are by looking in the **docs/dependencies** directory of the distribution, or by clicking here. The required dependencies are all the libraries listed in the **default** configuration.

IoC Container dependencies
As of Webwork 2.2, Spring is the recommended IoC container. If you plan to use Spring you will need the jars of the **spring** configuration. Other IOC containers are supported as well, but we recommend Spring as it is the most active and vibrant community

Below is a brief table of configurations and the functionality each will provide.

Dependency configuration functionalities

Configuration	Required to...
ajax	enable AJAX-related features in the UI tags
build	build WebWork from source
default	run the bare minimum support for WebWork
spring	use Spring integration
fileupload	support file uploads when the "jakarta" parser is selected in webwork.properties (recommended)
fileupload-cos	support file uploads when the "cos" parser is selected in webwork.properties
fileupload-pell	support file uploads when the "pell" parser is selected in webwork.properties
jasperreports	generate reports using JasperReports
jfree	generate reports using JFreeCharts
portlet	support for WebWork-enabled porlets
quickstart	start your applications using QuickStart
sitemesh	use FreeMarker- and Velocity-enabled decorators with SiteMesh
velocity	render results in Velocity
xslt	render results using XSLT

Dependency resolving in your Webwork based projects

Manual resolving

As stated previously, a look at the dependency page will provide you with the information on which jars you will need to include in your Webwork-based project.

Integrating Ivy to resolv dependencies

The much easier way for dependency resolving is to integrate Ivy into your project. See Building WebWork for a introduction to Ivy and installation instructions. Here is a sample ivy.xml for a Webwork-based project, requiring the latest release of Webwork 2.2 with freemarker, sitemesh and jasperreports functionality:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" xhref="http://www.jayasoft.fr/org/ivyrep/ivy-doc.xsl"?>
<ivy-module version="1.0">
    <info organisation="my.organisation.net" module="myproject"
          revision="1.0-alpha-1"
          status="integration"
          publication="20051022053520">
        <license name="Apache" url="http://www.apache.org/licenses/LICENSE-2.0.txt"/>
        <ivyauthor name="Me" url="http://my.organisation.net/"/>
        <description homepage="http://my.organisation.net/myproject">
            My first Webwork2 based project.
            <br/>
        </description>
    </info>
    <configurations>
        <conf name="build" visibility="private"/>
        <conf name="default"/>
    </configurations>
    <publications>
        <artifact name="myproject" type="jar" conf="default"/>
    </publications>
    <dependencies>
        <!-- build only dependencies -->
        <dependency org="junit" name="junit" rev="3.8.1" conf="build->*"/>
        <dependency org="servletapi" name="servletapi" rev="2.4" conf="build->*"/>

        <!-- runtime (and build) dependencies -->
        <dependency org="log4j" name="log4j" rev="1.2.9" conf="default->default"/>
        <dependency org="opensymphony" name="webwork" rev="2.2+"
conf="default->default,freemarker,sitemesh,jasperreports"/>

    </dependencies>
</ivy-module>
```

The following is a sample repository resolver configuration ivyconf.xml

```
<ivyconf>
    <properties file="ivyconf.properties"/>
    <conf defaultResolver="default" checkUpToDate="true"/>
    <resolvers>
        <ivyrep name="libraries"/>
        <chain name="default">
            <url name="opensymphony" checkmodified="true">
                <ivy
pattern="http://ivyrep.opensymphony.com/[organisation]/[module]/ivy-[revision].xml"/>
                <artifact
                        pattern="http://ivyrep.opensymphony.com/[organisation]/[module]/[artifact]-[revision
            </url>
            <url name="contegix">
                <ivy
```

```
"http://repository.contegix.com/ivyrep/[organisation]/[module]/ivy-[revision].xml"/>
                <artifact
                        pattern="http://repository.contegix.com/ivyrep/[organisation]/[module]/[artifact]-[r
            </url>
            <ivyrep name="ivyrep"/>
            <ibiblio name="contegix-maven" root="http://repository.contegix.com/maven"/>
            <url name="maven">
                <artifact
pattern="http://www.ibiblio.org/maven/[organisation]/jars/[module]-[revision].[type]"/>
            </url>
        </chain>
    </resolvers>
</ivyconf>
```

After integrating an appropriate Ivy init task into your project build file, Ivy will resolve **all** dependencies required by your project and download the needed jars. See Ivy documentation for more information on how to integrate Ivy in your own project, or just have a quick look in the Webwork2 build process.

EclipseWork

This page last changed on Mar 30, 2006 by scud.

EclipseWork is a third-party project created by Ricardo Lecheta. It is an Eclipse plugin for WebWork. Future plans include abstracting the plugin and providing for a plugin base that can be used to build both the Eclipse plugin and a new IDEA Plugin. For more information, check out the EclipseWork home page.

Examples

This page last changed on Mar 30, 2006 by scud.

- WW:Chat Application
- Hibernate's example app adminapp
- Hibernate's adminapp updated to use WW2.2, Hibernate3.1 and WW's new Spring2 IoC adminapp_new
.
- The current quartz sample web app also uses WW2 http://sourceforge.net/projects/quartz/
- AppFuse shows you how to get started quickly with WebWork, Spring and Hibernate.
- Equinox is a simpler version of AppFuse.
- EclipseWork's example app - Simple CRUD application WW2, Prevayler and Freemarker ww
- WW:SimpleLogin with Session

Asynchronous processing with WebWork - XWork

This page last changed on Mar 30, 2006 by scud.

This is from a presentation I gave at TheServerSide Java Symposium in March of 2005. Please don't use these materials for presentations without contacting me.

The Presentation

This presentation (in PowerPoint) covers why we'd want to take web request processing asynchronous, what problems it solves, and two methods for implementing asynchronous processing.

[Moving Web Apps From Synchronous to Asynchronous Processing|^async-web kt.ppt|Powerpoint presentation]

The Example App

The example app is a simple application showing the use of the "execAndWait" interceptor to provide the user an intermediate "in progress" page as well as a web request which is bridged into a JMS message to be handled asynchronously in a "fire and forget" mode.

[Example Application|^async.zip|Zipped app project]

Chat Application

This page last changed on Mar 30, 2006 by scud.

The following is a sample application I did to study how webwork works. Hopefully it'll help other newbies gain footing on this great framework. Having said that, if I have made some mistakes in this example, or if there is a better way of doing things, please feel free to contribute to this wiki.

The application works like a mini-BBS. Users login to the application with a nickname. The user session is saved in a session scoped component. Once logged in, they can leave quips or messages.

-

Basic Application and Environment Setup

webwork.properties

```
# Nothing here... that's right, it's empty. Using the webwork defaults.
```

-

Action Classes

LoginAction extends ActionSupport, which already implements some of the basic action methods. Additionally, it provides some validation support.

The two main methods we are concerned with in ActionSupport are validate() and execute().

(Note : this has changed from an earlier beta which uses doValidation() and doExecute().)

Validation

Validation is performed on your Action class if it implements Validatable (ActionSupport does), and your DefaultWorkflowInterceptor is activated on that action.

Execution

execute() returns a String. This String will be used to determine which result is used.

The framework provides some default return Strings, namely

```
Action.SUCCESS = "success"
Action.INPUT = "input"
Action.NONE = "none"
Action.ERROR = "error"
Action.LOGIN = "login"
```

For example, lets take a look at the relevant part of our xwork.xml configuration for LoginAction ...
xwork.xml

```
<action name="login" class="example.LoginAction">
  <result name="success" type="chain">
    <param name="actionName">viewMessages</param> </result>
  <result name="input" type="chain">
    <param name="actionName">viewMessages</param> </result>
</action>
```

If execute() returns a String of "success", the result with attribute "success" will be used. If doExecute() returns a String of "input", the result with attribute "success" will be used.

You can define your own return results. For example,

```
public String doExecute() {
  return "resetPassword";
}
```

```
<action name="login" class="example.LoginAction">
  <result name="resetPassword" type="chain">
    <param name="actionName">viewResetPassword</param> </result>
</action>
```

Context Variables / Mapping

LoginAction.java

```
public class LoginAction extends ActionSupport {

        private String loginName;

        public String getLoginName() {
                return loginName;
        }

        public void setLoginName(String loginName) {
                this.loginName = loginName;
        }
}
```

If you notice, login has a bean property loginName. This property will be set automatically by webwork from your web forms.

```
<form method="POST" action="login.action">
  <input type="text" name="loginName" size="20">
  <input type="submit" value="Login">
</form>
```

Also, the bean property is available to your views. In velocity, this accessible via the VelocityContext

```
$loginName
```

which is mapped to getLoginName() from your Action class.

getLoginName() is mapped to $loginName. You can map any other object you wish. For example, I could have a User object, and thus...

```
class User {
   private Account account;
   private String name;
   // with the relevant getX() methods...
}
```

in my action class...

```
class MyExampleAction {
   //...
   User getUser() {
      returns user;
   }
   //...
}
```

and in my velocity template, have a

```
Welcome, $user.name.
You last logged on at $user.lastLogin.
You currently have $user.account.balance left in your account.
```

-

Result Types

Predefined in webwork-default.xml

```
<result-types>
            <result-type name="dispatcher"
class="com.opensymphony.webwork.dispatcher.ServletDispatcherResult"/>
            <result-type name="redirect"
class="com.opensymphony.webwork.dispatcher.ServletRedirectResult"/>
            <result-type name="velocity"
class="com.opensymphony.webwork.dispatcher.VelocityResult"/>
            <result-type name="chain" class="com.opensymphony.xwork.ActionChainResult"/>
        </result-types>
```

In our example, we have only used the results of <chain> and <velocity>.

-

Interceptors

is there an order execution of the interceptors ? Which interceptor is executed first ?

Interceptors are called before the actions are invoked. With interceptors, you can "wrap" your action classes to provide additional services or functions. You can even prevent the execution of the action classes if you wish.

Some interceptors are predefined in webwork-default.xml

```
<interceptors>
            <interceptor name="component"
class="com.opensymphony.xwork.interceptor.component.ComponentInterceptor"/>
            <interceptor name="validation"
class="com.opensymphony.xwork.validator.ValidationInterceptor"/>
            <interceptor name="workflow"
class="com.opensymphony.xwork.interceptor.DefaultWorkflowInterceptor"/>
            <interceptor-stack name="defaultStack">
                <interceptor-ref name="timer"/>
                <interceptor-ref name="logger"/>
                <interceptor-ref name="static-params"/>
                <interceptor-ref name="params"/>
            </interceptor-stack>
        </interceptors>
```

... more info on interceptor-ref

... more info on interceptor-stack

-

Components

components.xml

```
<components>

    <component>
      <scope>application</scope>
      <class>example.data.MessageList</class>
      <enabler>example.data.MessageListAware</enabler>
    </component>

    <component>
      <scope>session</scope>
      <class>example.web.UserSession</class>
      <enabler>example.web.UserSessionAware</enabler>
    </component>

  </components>
```

Two components, one to hold the application scoped chat messages, another to hold the user's session information (login account name, etc.) For all practical purposes, you can replace the application scoped component with a database. i.e. instead of reading/writing to the component, read/write to the database.

-

Comments

Feedback welcome. I'm also a newbie, and may have misused the framework, coded some mistakes above. Apologies if that be the case.

SimpleLogin with Session

This page last changed on Mar 30, 2006 by scud.

I wrote this simple application to demostrate how to use webwork in a login/logout.

/Login.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
 pageEncoding="ISO-8859-1"%>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head><body>
<form action="login.action" method="post">
User id<input type="text" name="userId" /> <br/>
Password <input type="password" name="passwd" /> <br />
<input type="submit" value="Login"/>
</form>
</body>

</html>
```

/pages/welcome.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib prefix="ww" uri="/webwork" %>
<jsp:include page="WEB-INF/inc/loginCheck.jsp" />
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Welcome</title>
</head>

<body>Welcome, you have logined. <br />
The attribute of 'context' in session is
<ww:property value="#session.context" />
<br /><br /><br />
<a xhref="<%= request.getContextPath() %>/logout.action">Logout</a>
<br />
<a xhref="<%= request.getContextPath() %>/logout2.action">Logout2</a>
</body>
</html>
```

/WEB-INF/inc/loginCheck.jsp

```
 <%@ taglib="/webwork" prefix="ww" %>
<ww:if test="#session.login != 'true'">
<jsp:forward page="<%= request.getContextPath() %>/login.jsp" />
</ww:if>
```

simple.LoginAction.java

```
package simple;
import java.util.Date;import java.util.Map;

import javax.servlet.http.HttpSession;

import com.opensymphony.webwork.ServletActionContext;
import com.opensymphony.xwork.ActionSupport;

public class LoginAction extends ActionSupport {

    private String userId;
    private String passwd;

    public String execute() throws Exception {
        if ("admin".equals(userId) && "password".equals(passwd)) {
//            HttpSession session = ServletActionContext.getRequest().getSession();
//            session.setAttribute("logined","true");
//            session.setAttribute("context", new Date());
// Better is using ActionContext
  Map session = ActionContext.getContext().getSession();
session.put("logined","true");
            session.put("context", new Date());
            return SUCCESS;
        }
        return ERROR;
    }

    public String logout() throws Exception {
//        HttpSession session = ServletActionContext.getRequest().getSession();
//        session.removeAttribute("logined");
//        session.removeAttribute("context");
 Map session = ActionContext.getContext().getSession();
 session.remove("logined");
        session.remove("context");
        return SUCCESS;
    }

    public String getPasswd() {
        return passwd;
    }

    public void setPasswd(String passwd) {
        this.passwd = passwd;
    }

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }
}
```

simple.LogoutAction.java

```
package simple;
import java.util.Map;
import javax.servlet.http.HttpSession;

import com.opensymphony.webwork.ServletActionContext;
import com.opensymphony.xwork.ActionSupport;

public class LogoutAction extends ActionSupport {

    public String execute() throws Exception {
     Map session = ActionContext.getContext().getSession();
session.remove("logined");
session.remove("context");
        return SUCCESS;
    }

}
```

/WEB-INF/classes/xwork.xml

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.1.1//EN"
"http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">

<xwork>
    <include file="webwork-default.xml"/>

    <package name="default" extends="webwork-default">
        <!-- Add your actions here -->
        <action name="login" class="simple.LoginAction" >
            <result name="success" type="dispatcher">/pages/welcome.jsp</result>
            <result name="error" type="redirect">/login.jsp</result>
        </action>

        <action name="logout2" class="simple.LoginAction" method="logout" >
            <result name="success" type="redirect">/login.jsp</result>
        </action>

        <action name="logout" class="simple.LogoutAction" >
            <result name="success" type="redirect">/login.jsp</result>
        </action>
    </package>
</xwork>
```

Exception Handling

This page last changed on Apr 29, 2006 by scud.

综述

异常映射 是一个处理抛出异常的Action的强大特性. 这个思想的核心是捕捉到Action的方法执行期间抛出的异常,并把它映射到一个result, 既可以是全局的也可以是action作用域内的results. 这对框架尤其有用, 比如Hibernate和Acegisecurity抛出的RuntimeExceptions.

为了和WebWork的其他部分协作,开启异常映射功能需要一个拦截器. 下面的代码片断是来自webwork-default.xml的,其中已经启用了异常映射.

```
...
<interceptors>
    ...
    <interceptor name="exception"
class="com.opensymphony.xwork.interceptor.ExceptionMappingInterceptor"/>
    ...
</interceptors>

<!-- Basic stack -->
<interceptor-stack name="basicStack">
    <interceptor-ref name="exception"/>
    <interceptor-ref name="servlet-config"/>
    <interceptor-ref name="prepare"/>
    <interceptor-ref name="static-params"/>
    <interceptor-ref name="params"/>
    <interceptor-ref name="conversionError"/>
</interceptor-stack>
...
```

异常处理的下一步就是映射实际的异常到特定的result.WebWork提供了两种方式来声明一个异常映射 `<exception-mapping/>` - 全局的或者针对一个特定的action. 异常映射元素有两个属性,`exception` 和 `result`.

当声明一个异常映射时,拦截器会根据抛出的异常来查找声明中的层次关系最接近的类.拦截器会检查此Action所有可用的声明的映射,包括Action特定的和全局的映射.result(全局或者Action范围的)然后会被调用.

这遵从一个Action返回一个result的一般规则. 它首先查看Action中的result定义, 当没有找到时查看全局的result定义.

下面的例子是一个全局和Action作用域内的异常映射的例子.

```
<xwork>
    <package name="default">
        ...
        <global-results>
            <result name="login" type="redirect">/login.action</result>
            <result name="rootException"
type="freemarker">/WEB-INF/views/exception.ftl</result>
        </global-results>

        <global-exception-mappings>
            <exception-mapping exception="java.sql.SQLException" result="sqlException"/>
            <exception-mapping exception="java.lang.Exception" result="rootException"/>
        </global-exception-mappings>
        ...
        <action name="myAction" class="...">
            <interceptor-ref name="exception" />
            <exception-mapping exception="com.acme.foo.SecurityException" result="login"/>
            <result name="sqlException" type="chain">sqlExceptionAction</result>
            <result name="success" type="freemarker">/WEB-INF/views/acme/success.ftl</result>
        </action>
        ...
    </package>
</xwork>
```

在上面的例子中，基于每一个异常的发生处理如下:

- 一个 `java.sql.SQLException` 将链接到 `sqlExceptionAction`
- 一个 `com.acme.foo.SecurityException` 将转向到 `/login.action`
- 任何其他继承于 `java.lang.Exception` 的异常将执行FreeMarker类型的result rootException ，显示页面 `/WEB-INF/views/exception.ftl`

ValueStack中的异常值(Exception Values on the ValueStack)

缺省情况下异常映射拦截器(ExceptionMappingInterceptor)添加下列值到Value Stack中:

- exception - 异常对象自己
- exceptionStack - 异常堆栈

FAQ

This page last changed on Mar 30, 2006 by scud.

Each question should be a new page. Typically answers should link to content in the reference documentation. If the answer isn't in the reference, then it should probably be added there and then linked to from the FAQ. Also note that some of the questions are current'y too verbose and should be broken down given that they have already been categorized (ie: Validation, Internationlization, etc).

General

- How do I get the latest version of WebWork
- What are the default variables in the value stack
- How do I get access to the session
- How can I see all request parameters passed into the action
- How can I get the HttpServletRequest
- How can I get the HttpServletResponse
- Can I break up my large XWork.xml file into smaller pieces
- I'm trying to run the webwork example in the tutorial on Tomcat, and it can't instantiate the VelocityEngine
- How do I handle files upload
- How do I get JEE J2EE security info
- How do I get static parameters into my action
- Can I access my action's Result
- Can I enable ww altSyntax on a per\-page basis
- Can I change theme on a per\-page basis
- Can I change templateDir on a per\-page basis
- Can I change templateSuffix on a per\-page basis
- How can I display image that are contained as bytes in my action
- How to reload xwork configuration
- Why does webwork keeps calling my action's properties \(eg. my getModel if my action implements ModelDriven\) multiple times
- How to reload xwork configuration

Tags

- How can I put a String literal in a Javascript call, for instance in an onChange attribute
- Why won't the 'if' tag evaluate a one char string
- Why does FreeMarker complains that there's an error in my user\-directive when I used JSP Tag
- Can I have my webwork action tag run another method apart from the default execute method

Inversion of Control

- How can I integrate WebWork IoC in to an object that is not an action

Validation

- How do I use messages from within the validator
- Why do I get error message saying Attribute 'short\-circuit' must be declared for element type 'field\-validator'
- How do I populate my action properties upon validation failure

Internationalization

- How do I set a global resource bundle
- How do I decouple XWork LocalizedTextUtil global resource bundle loading from serlvets

- How do I add I18N to a UI tag, like the textfield tag
- Can I add I18N outside the Action's context
- How to support UTF\-8 URIEncoding with Tomcat
- How to escape special chars in resource bundles

Type Conversion

- How do I change the error message for invalid inputted fields
- Why am I getting RuntimeException saying Compound Root cannot find a particular Object with a particular property

Interceptor

- Why isn't my Prepare interceptor being executed

Portlet Support (JSR168)

- Which portal servers are supported
- How to build the portlet war for a specific portal server

Can I access my action's Result

This page last changed on Mar 30, 2006 by scud.

Yes, that could be done with the help of an interceptor.

+ Method One +

```
public class MyInterceptor implements Interceptor {
  ...
        public String intercept(ActionInvocation invocation) throws Exception {
                Map<String, ResultConfig> resultsMap =
invocation.getProxy().getConfig().getResults();

                // do something with ResultConfig in map

                return invocation.invoke();
        }
  ...
}
```

+ Method Two +

```
public class MyInterceptor implements Interceptor {
  ...
        public String intercept(ActionInvocation invocation) throws Exception {
             invocation.addPreResultListener(new PreResultListener() {
                        public void beforeResult(ActionInvocation invocation, String
resultCode) {
                                Map<String, ResultConfig> resultsMap =
invocation.getProxy().getConfig().getResults();
                                ResultConfig finalResultConfig = resultsMap.get(resultCode);

                                // do something interesting with the 'to-be' executed result
                 }
             });

             return invocation.invoke();
        }

  ...
}
```

The difference between Method One and Two is that method two gives one, the final result to be executed.

Can I add I18N outside the Action's context

This page last changed on Mar 30, 2006 by scud.

Yes, use the <ww:i18n> tag to push a resource bundle on to the stack. Now calls with <ww:text/> or <ww:property value="getText(...)"/> will read from that resource bundle.

Can I break up my large XWork.xml file into smaller pieces

This page last changed on Mar 30, 2006 by scud.

Sure, that's what the <include> element is for. Most xwork.xml files already have one:

```
<xwork>
    <include file="webwork-default.xml"/>
    <include file="config-browser.xml"/>
    <package name="default" extends="webwork-default">
....
    </package>
    <include file="other.xml"/>
</xwork>
```

This tells it to load the webwork-default.xml from the webwork jar file to get all of those interceptor and result definitions.

You can put your own <include> in your xwork.xml interchangeably with <package> elements... They will be loaded in the same order as it reads from top to bottom and adds things as it reads them.

@see com.opensymphony.xwork.config.ConfigurationManager
@see com.opensymphony.xwork.config.Configuration
@see com.opensymphony.xwork.config.impl.DefaultConfiguration
@see com.opensymphony.xwork.config.ConfigurationProvider
@see com.opensymphony.xwork.config.providers.XmlConfigurationProvider

Can I change templateDir on a per-page basis

This page last changed on Mar 30, 2006 by scud.

Yes, by using the `<ww:set name="templateDir" value="myTemplateDir" scope="page" />`.

> ⚠ This will set the attribute 'templateDir' in the page scope, with the value of what that is returned by 'myTemplateDir' in the stack, eg. the action class might have a getMyTheme method that return a String called template, to indicate what the template directory is within the classpath.

Another way would be `<ww:set name="templateDir" value="'template'" scope="page" />`

> ⚠ This will set the attribute 'templateDir' in the page scope with the value 'template' to indicate what the template directory is within the classpath.

See Template Loading

Can I change templateSuffix on a per-page basis

This page last changed on Mar 30, 2006 by scud.

Yes, by using the `<ww:set name="templateSuffix" value="'vm'" />` see Template Loading.

Can I change theme on a per-page basis

This page last changed on Mar 30, 2006 by scud.

Yes, by using the `<ww:set name="theme" value="myTheme" scope="page" />`.

> ⚠ This will set the attribute 'theme' in the page scope, with the value of what that is returned by 'myTheme' in the stack, eg. the action class might have a getMyTheme method that return a String called simple, to indicate simple theme.

Another way would be `<ww:set name="theme" value="'simple'" scope="page" />`

> ⚠ This will set the attribute 'theme' in the page scope with the value 'simple' to indicate a simple theme.

See Selecting Themes

Can I enable ww altSyntax on a per-page basis

This page last changed on Mar 30, 2006 by scud.

Yes, by using the `<ww:set name="useAltSyntax" value="true" />` see Alt Syntax.

Can I have my webwork action tag run another method apart from the default execute method

This page last changed on Mar 30, 2006 by scud.

Yes that could be done, eg if one like the method to be executed is input(), the following could be applied

```
<ww:action name="myActionAlias!input" .... />
```

How can I display image that are contained as bytes in my action

This page last changed on Mar 30, 2006 by scud.

With the html as follows:

```
<img src="/myWebAppContext/myAction!default.action />
```

xwork.xml could be as follows:

```
<xwork>
    ...
    <result-types>
        <result-type name="myBytesResult" class="foo.bar.MyBytesResult" />
    </result-types>
    ...

    <action name="myAction" class="...">
        <result name="myImageResult" type="myBytesResult">
            <param name="contentType">${myContentType}</param>
            <param name="contentDisposition">${myContentDisposition}</param>
            <param name="contentLength">${myContentLength}</param>
            <param name="bufferSize">${myBufferSize}</param>
        <result>
    </action>

    ...
</xwork>
```

the action could be as follows:

```
public class MyAction extends ActionSupport {
   public String doDefault() {
        return "myImageResult";
    }
   public byte[] getMyImageInBytes() { .... }

   public String getMyContentType() { ... }
   public String getMyContentDisposition() { ... }
   public int getMyContentLength() { .... }
   public int getMyBufferSize() { ... }
}
```

```
public class MyBytesResult implements Result {

        public void execute(ActionInvocation invocation) throws Exception {
                MyAction action = (MyAction) invocation.getAction();
                HttpServletResponse response = ServletActionContext.getResponse();

                response.setContentType(action.getContentType());
                response.setContentLength(action.getContentLength());

                response.getOutputStream().write(action.getImageInBytes());
                response.getOutputStream().flush();
        }

}
```

How can I get the HttpServletRequest

This page last changed on May 19, 2006 by scud.

Method A:
ServletActionContext.getRequest() (works internally using a ThreadLocal)

Method B:
Have the action implements ServletRequestAware and the servlet request will be set through setServletRequest(HttpServletRequest) method. This requires the action to have a 'servlet-config' interceptor added.

@see webwork\-default.xml
@see com.opensymphony.webwork.interceptor.ServletRequestAware
@see com.opensymphony.webwork.interceptor.ServletConfigInterceptor

How can I get the HttpServletResponse

This page last changed on May 19, 2006 by scud.

Method A:
ServletActionContext.getResponse() (works internally using a ThreadLocal)

Method B:
Have the action implements ResponseAware and the response will be set through setServletResponse(HttpServletResponse). The action needs to have 'servlet-config' interceptor added to it.

@see webwork\-default.xml
@see com.opensymphony.webwork.interceptor.ServletResponseAware
@see com.opensymphony.webwork.interceptor.ServletConfigInterceptor

How can I integrate WebWork IoC in to an object that is not an action

This page last changed on Mar 30, 2006 by scud.

Obtain the ComponentManager from the request: ComponentManager cm = (ComponentManager) ServletActionContext.getRequest().getAttribute("DefaultComponentManager");
then you need to initialize it using: cm.initializeObject(Object)

How can I put a String literal in a Javascript call, for instance in an onChange attribute

This page last changed on Mar 30, 2006 by scud.

The problem is in escaping quotes and getting the double quotes around the final value, like we expect in HTML attributes. Here's an example of the right way to do this (thanks to John Brad):

```
onchange='"someFunc(this.form, \'abc\')"'
```

Notice here that there are single quotes surrounding the double quotes, and then the single quotes inline in the Javascript are escaped. This produces this result:

```
onchange="someFunc(this.form, 'abc')"
```

How can I see all request parameters passed into the action

This page last changed on May 19, 2006 by scud.

Method A:
ActionContext.getContext().getParameters() (returns Map, works internally using a ThreadLocal)

Method B:
Have the action implements ParameterAware interface and the parameters will be set through the setParameters(Map) method. This requires that the 'servlet-config' interceptor being added to that particular action.

@see webwork\-default.xml
@see com.opensymphony.webwork.interceptor.ParameterAware
@see com.opensymphony.webwork.interceptor.Servlet Config Interceptor

How do I add I18N to a UI tag, like the textfield tag

This page last changed on Mar 30, 2006 by scud.

```
<ww:textfield label="%{getText('i18n.label')}" name="label1" />
```

This will get the localized text message for the key "i18n.key" and put it in the label.

Alternatively, portion of controlheader-core.ftl in /template/xhtml could be modified (if xhtml theme is being used), as follows :-

```
${parameters.label?html}:<#t/>
```

```
<#assign mm="getText('"+parameters.label?html+"')" /><#t/>
${stack.findValue(mm)}:<#t/>
```

or

```
${stack.findValue("getText('"+parameters.label?html+"')")}
```

such that using text tag like following will work as well (such that the label printed will be i18n).

```
<ww:textfield label="i18n.label" name="label1" />
```

How do I change the error message for invalid inputted fields

This page last changed on Mar 30, 2006 by scud.

You need to create a message for that field, for example if you have a user.dob field you would use this in your messages file (see above for example on setting a global messages file):
invalid.fieldvalue.user.dob=Please enter Date of Birth in the correct format.

How do I decouple XWork LocalizedTextUtil global resource bundle loading from serlvets

This page last changed on Mar 30, 2006 by scud.

If you're using XWork outside a Web context, then use whatever startup hooks you have in that context (i.e. application start for a desktop app) to add the global resource bundle. This is a startup activity, so use whatever mechanisms are provided in the context you're running in.

How do I get access to the session

This page last changed on May 19, 2006 by scud.

Method A:
ActionContext.getContext().getSession() (returns Map, works internally using a ThreadLocal)

Method B (Recommended):
Have the action implements SessionAware, and the Session (as a Map) will be set through the setSession(Map) method. This requires that the 'servlet-config' interceptor being included when the particular action is processed.

@see webwork\-default.xml
@see com.opensymphony.webwork.interceptor.SessionAware
@see com.opensymphony.webwork.interceptor.Servlet Config Interceptor

How do I get JEE J2EE security info

This page last changed on Mar 30, 2006 by scud.

Method A:

```
HttpServletRequest request = ....   // get HttpServletRequest
request.getAuthType()               // http or https
request.getRemoteUser()             // the user principal (in string)
request.getUserPrincipal()          // get a Principal object
request.isUserInRole(String)
```

Method B: (Recommended)

- Not tied to Servlet spec
- Help in unit testing

Have the action implements PrincipalAware and add 'servlet-config' interceptor to it. a PrincipalProxy object will be set to method setPrincipalProxy(PrincipalProxy). With PrincipalProxy, one could have access to methods such as isUserInRole(), getUserPrincipal(), getRemoteUser(), isRequestSecure() etc.

@see com.opensymphony.webwork.interceptor.PrincipalProxy
@see com.opensymphony.webwork.interceptor.PrincipalAware
@see com.opensymphony.webwork.interceptor.ServletConfigInterceptor

How do I get static parameters into my action

This page last changed on Mar 30, 2006 by scud.

Static parameters could be defined into an action through xwork.xml like bellow:

```
<action name="myAction" class=" ... ">
     <param name="myStaticParam1">myStaticValue1</param>
     <param name="myStaticParam2">myStaticValue2</param>
     <param name="myStaticParam3">myStaticValue3</param>
  </action>
```

+ Method A +
Have the action class itself implements com.opensymphony.xwork.config.entities.Parameterizable and the static parameters will be set into it through the setParams(Map) method. In the example above, the key and value will be as tabulated below:

key	value
myStaticParam1	myStaticValue1
myStaticParam2	myStaticValue2
myStaticParam3	myStaticValue3

+ Method B +
Have the action class itself define getter/setter for the static parameter itself and those static parameter will be set through those setter and getter. In the case above, the action class could be as follows:

```
public class MyAction extends ActionSupport {
     ...

     public String getMyStaticParam1() { ....}
     public void setMyStaticParam1(String myStaticParam1) { ... }

     public String getMyStaticParam2() { ... }
     public void setMyStaticParam2(String myStaticParam2) { ... }

     public String getMyStaticParam3() { ... }
     public void setMyStaticParam3(String myStaticParam3) { ... }

     ...
  }
```

The getter and setter, will be set appropriately.

NOTE: For this to work, 'static-params' interceptor must be added to the action.

@see com.opensymphony.xwork.interceptor.StaticParametersInterceptor
@see com.opensymphony.xwork.config.entities.Parameterizable

How do I get the latest version of WebWork

This page last changed on Mar 30, 2006 by scud.

Distibution package

The latest distribution packages including official beta versions are found on Java DevNet.

From CVS (The Bleeding Edge)

Short answer:

```
cvs -d :pserver:guest@cvs.dev.java.net:/cvs login
(Use an empty password, just hit enter..)
cvs -d :pserver:guest@cvs.dev.java.net:/cvs checkout opensymphony
cvs -d :pserver:guest@cvs.dev.java.net:/cvs checkout webwork
```

optional:

```
cvs -d :pserver:guest@cvs.dev.java.net:/cvs checkout xwork
```

Note: One needs to have ivy in $ANT_HOME/lib, cause WebWork uses ivy to manage its library dependencies.

Long answer:
See Building Webwork for a detailed description, including information on Ivy and JDK compatibility.

How do I handle files upload

This page last changed on Mar 30, 2006 by scud.

Method A
MultipartRequestWrapper multipartRequest = ((MultipartRequestWrapper)ServletActionContext.getRequest())

With multipartRequest, one could access methods such as getFiles(...), getFile(...), getContentType(...), hasErrors(), getErrors() etc to handle the file uploaded.

Method B (Recommended)
Add a 'fileUpload' interceptor to the action. For example, in the following case:

```
<form name="myForm" enctype="multipart/form-data">
     <input type="file" name="myDoc" value="Browse ..." />
     <input type="submit" />
  </form>
```

The action class would requires any (or none, but if none what is the point?) of three methods being defined, in order for the interceptor to populate it with uploaded file information

```
public void setMyDoc(File myDoc) { ...}
public void setMyDocContentType(String contentType) { .... }
public void setMyDocFileName(String filename) { .... }
```

with these methods, one could do whatever is needed with the uploaded file. If multiple files are uploaded as in following:

```
<form name="myForm" enctype="multipart/form-data">
      <input type="file" name="myDoc" value="Browse File A ..." />
      <input type="file" name="myDoc" value="Browse File B ..." />
      <input type="file" name="myDoc" value="Browse File C ..." />
      <input type="submit" />
   </form>
```

The action class needs only make the corresponding method an array, orders followed such that getMyDoc()0 will have its content type as getMyDoc()0 and its file name as getMyDoc()1.

```
public void setMyDoc(File[] myDocs) { ... }
public void setMyDocContentType(String[] contentTypes) { ... }
public void setMyDocFileName(String[] fileNames) { ... }
```

Extra Information:
The following properties in webwork.properties affect the file upload.

webwork.multipart.parser (as of WW2.2 its jakarta by default)
webwork.multipart.saveDir (default to javax.servlet.context.tempdir defined by container)
webwork.multipart.maxSize (approximately 2M by default)

@see webwork.properties
@see com.opensymphony.webwork.dispatcher.FilterDispatcher#doFilter(SerlvetRequest, ServletRepsonse, FilterChain)
@see com.opensymphony.webwork.dispatcher.DispatcherUtil#wrapRequest(HttpServletRequest, SerlvetContext)
@see com.opensymphony.webwork.dispatcher.multipart.MultipartRequestWrapper

@see com.opensymphony.webwork.interceptor.FileUploadInterceptor

How do I populate my action properties upon validation failure

This page last changed on Mar 30, 2006 by scud.

Say for example we have a

```
&lt;ww:checkboxlist
      name="selectedOptions"
      list="options"
      listKey="id"
      listValue="name" /&gt;
```

If we like to populate options upon validation failure, we could have a prepare interceptor above the validation interceptor in the interceptor stack.

```
....
   public void prepare() throws Exception {
      // populate the options property list with options
      // that are supposed to be checked.
   }
   ....
```

How do I set a global resource bundle

This page last changed on Mar 30, 2006 by scud.

In webwork.properties(as of Webwork 2.1.1),
you can now use:

```
webwork.custom.i18n.resources=global-messages
```

Serveral resource bundles can be specified by comma separating them.
for example see webwork.properties :
http://wiki.opensymphony.com/display/WW/webwork.properties

Java class (thanks Drew McAuliffe):

```
public class WebworkGlobalMessagesListener implements ServletContextListener {
    private static Logger log = Logger.getLogger(WebworkGlobalMessagesListener.class);
    private static final String DEFAULT_RESOURCE = "global-messages";

    /**
     * Uses the LocalizedTextUtil to load messages from the global
     message bundle.
     * @see
     javax.servlet.ServletContextListener#contextInitialized(javax.servlet.Servle
     tContextEvent)
     */
    public void contextInitialized(ServletContextEvent arg0) {
        log.info("Loading global messages from " + DEFAULT_RESOURCE);
        LocalizedTextUtil.addDefaultResourceBundle(DEFAULT_RESOURCE);
        log.info("Global messages loaded.");
    }

    /**
     * @see
javax.servlet.ServletContextListener#contextDestroyed(javax.servlet.ServletContextEvent)
     */
    public void contextDestroyed(ServletContextEvent arg0) {

        // do nothing
    }

}
```

web.xml:
(under listeners section)

```
<listener>
<listener-class>mypackagename.WebworkGlobalMessagesListener</listener-class>
</listener>
```

How do I use messages from within the validator

This page last changed on Mar 30, 2006 by scud.

```
<validators>
    <field name="name">
        <field-validator type="requiredstring">
            <message key="template.name.errors.required">A default message in case the key is
not found</message>
        </field-validator>
    </field>
</validators>
```

How to build the portlet war for a specific portal server

This page last changed on Mar 30, 2006 by scud.

Building the portlet sample webapp for a specific container

Build instructions

Some text with a title

- run 'ant' from the webapp project dir
- cd to webapps directory
- To build a portlet webapp, simply run: ant build-portlet

- You can use the 'container' system property to target a specific container, e.g. '-Dcontainer=exo'.
  Supported containers are exo, gridsphere, liferay-3.6.1, jboss-portal-2.0, jboss-portal-2.2 and jetspeed2.
  For IBM WebSphere Portal 5.1, no container property is required.
  For example:
  ant build-portlet -Dcontainer=jboss-portal-2.2
- Check the etc/yourcontainer directory for a Readme. If available, check for further instructions.
- deploy the war to your portal server

How to escape special chars in resource bundles

This page last changed on Mar 30, 2006 by scud.

Normal Java resource bundles

In this case the important aspect is the following:

> **API: java.util.Properties**
> The method does not treat a backslash character, \, before a non-valid escape character as an error; the backslash is silently dropped. For example, in a Java string the sequence "\z" would cause a compile time error. In contrast, this method silently drops the backslash. Therefore, this method treats the two character sequence "\b" as equivalent to the single character 'b'.

MessageFormat rules

Extensively describing rules for embedding ' and

Unknown macro: {. (see javadoc API for MessageFormat) The special chars ', { and }}
with ': resulting in '}'

- enclose { with ': resulting in '{'

How to reload xwork configuration

This page last changed on Mar 30, 2006 by scud.

If you are finding that the xwork configuration is not reloaded when you redeploy your war. Is there a way to tell webwork to unload its configuration when the context is destroyed.

Try to call

```
com.opensymphony.xwork.config.ConfigurationManager.destroyConfiguration()
```

Useful Information

This should destroy the current configuration and perform a config reload on next **request**. You can use a action for do it.

another tip

This is another option that you can use, change for true.

```
### Configuration reloading
### This will cause the configuration to reload xwork.xml when it is changed
webwork.configuration.xml.reload=false
```

How to support UTF-8 URIEncoding with Tomcat

This page last changed on Mar 30, 2006 by scud.

If your POST and GET parameters are not UTF-8 encoded when using Tomcat 5.x, try to adjust the Connector configuration in Tomcats server.xml like this:

```
<!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
   <Connector port="8080" maxHttpHeaderSize="8192"
               maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
               enableLookups="false" redirectPort="8443" acceptCount="100"
               connectionTimeout="20000" disableUploadTimeout="true"
               URIEncoding="UTF-8"
   />
```

I'm trying to run the webwork example in the tutorial on Tomcat, and it can't instantiate the VelocityEngine

This page last changed on Mar 30, 2006 by scud.

Tomcat says:

```
javax.servlet.ServletException: Servlet.init() for servlet webwork threw exception at
org.apache.catalina.core.StandardWrapper.loadServlet(StandardWrapper.java:963)
...
root cause


java.lang.RuntimeException: Unable to instantiate VelocityEngine!
at
com.opensymphony.webwork.views.velocity.VelocityManager.newVelocityEngine(VelocityManager.java:333)
at
com.opensymphony.webwork.views.velocity.VelocityManager.init(VelocityManager.java:146)
at
com.opensymphony.webwork.dispatcher.ServletDispatcher.init(ServletDispatcher.java:177)
at
org.apache.catalina.core.StandardWrapper.loadServlet(StandardWrapper.java:935)
```

Solution: (thanks to Keith Lea)

It turns out Velocity's Avalon logging system was trying to write to my tomcat folder.

So that it's on file somewhere for other people, I will describe the solution:

I created a file "velocity.properties" and placed it in my WEB-INF/classes folder. Inside the file I wrote:

```
runtime.log.logsystem.class=org.apache.velocity.runtime.log.NullLogSystem
```

This stops velocity from logging, and makes webwork work again.

What are the default variables in the value stack

This page last changed on Mar 30, 2006 by scud.

Variables	Description
attr	scans the request, session, and application attributes, in that order
request	request attributes
session	session attributes
application	application attributes
parameters	request params

Defining Java Constant	Variables	Description
ActionContext.PARAMETERS	com.opensymphony.xwork.ActionContext.parameters	Same as ’parameters’ above
ActionContext.SESSION	com.opensymphony.xwork.ActionContext.session	Same as ’session’ above
ActionContext.APPLICATION	com.opensymphony.xwork.ActionContext.application	Same as ’application’ above
ActionContext.LOCALE	com.opensymphony.xwork.ActionContext.locale	From locale defined in webwork.properties else from the request object
ActionContext.DEV_MODE	__devMode	true or false if in development mode or otherwise whereby resource bundle webwork.properties, xwork.xml, converters, validators will be refreshed when changes
ActionContext.HTTP_REQUEST	com.opensymphony.xwork.dispatcher.HttpServletRequest	Same as ’request’ above
ActionContext.HTTP_RESPONSE	com.opensymphony.xwork.dispatcher.HttpServletResponse	Same as ’response’ above
ActionContext.SERVLET_CONTEXT	com.opensymphony.xwork.dispatcher.ServletContext	Servlet spec’s ServletContext object
ActionContext.COMPONENT_MANAGER	com.opensymphony.xwork.interceptor.component.ComponentManager	Webwork’s IoC component manager

For further information
@see com.opensymphony.webwork.dispatcher.DispatcherUtils#createContextMap
@see com.opensymphony.xwork.interceptor.component.ComponentInterceptor
@see com.opensymphony.webwork.WebWorkStatics

Which portal servers are supported

This page last changed on Mar 30, 2006 by scud.

List of supported portal servers

The following portal servers are known to work with the webwork portlet integration:

- eXo Portal 1.1 http://www.exoplatform.com/portal/faces/public/exo
- Gridsphere 2.1 http://www.gridsphere.org/gridsphere/gridsphere
- Liferay-3.6.1 http://liferay.com
- JBoss-Portal 2.0 http://www.jboss.com/products/jbossportal
- JBoss-Portal 2.2 http://www.jboss.com/products/jbossportal
- Apache Jetspeed 2 http://portals.apache.org/jetspeed-2/
- Pluto 1.0.1 http://portals.apache.org/pluto/
- IBM WebSphere Portal 5.1

If your portal server is not listed here and you would like to contribute required config files and additional deployment descriptors, please add them to Jira. http://jira.opensymphony.com/browse/WW

Why am I getting RuntimeException saying Compound Root cannot find a particular Object with a particular property

This page last changed on Mar 30, 2006 by scud.

This exception could be thrown if WebWork's development mode is turn on. It is done through webwork.properties.

```
webwork.devMode = true
```

Why didn't my webwork action tag gets executed when I have validation errors

This page last changed on Mar 30, 2006 by scud.

WebWork action tag will not get executed when there's a validation error with ValidationInterceptor and DefaultWorkflowInterceptor in place.

One way to make the action gets executed if to have the action's execution method specified in ValidationInterceptor's excludeMethods parameter as show in the following snippet.

```
<interceptor-ref name="validation">
  <param name="excludeMethods">input,back,cancel,browse</param>
</interceptor-ref>
```

and defined the method on the action to be executed as one of those excludedMethods, eg.

```
<ww:action name="myActionAlias!input" executeResult="false" />
```

Why do I get error message saying Attribute 'short-circuit' must be declared for element type 'field-validator'

This page last changed on Mar 30, 2006 by scud.

The DTD needs to be of version 1.0.2 instead of 1.0, as the short-circuit is introduced in the 1.0.2 version

```
<!DOCTYPE validators PUBLIC
                "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
                 "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
```

Why does FreeMarker complains that there's an error in my user-directive when I used JSP Tag

This page last changed on Mar 30, 2006 by scud.

To use JSP Tags in FreeMarker template, the following needs to be included in the web.xml

```
<servlet>
    <servlet-name>jspSupportServlet</servlet-name>
    <servlet-class>com.opensymphony.webwork.views.JspSupportServlet.JspSupportServlet</servlet-class>
    <load-on-startup>10</load-on-startup>
</servlet>
```

The snippets above, register a JspSupportServlet with the webapp, and requires that the container load it upon startup. It provides access to the servlet instance itself where valuable information like ServletContext could be obtained. This is needed for FreeMarker template rendering that contains JSP tags.

Why does webwork keeps calling my action's properties (eg. my getModel if my action implements ModelDriven) multiple times

This page last changed on Mar 30, 2006 by scud.

This is due to OGNL's accessor calls. One possible improvement would be to have the action do some caching if possible. Say if getModel is called multiple times it might be worthwhile considering the possibilities of caching the returned model itself.

Why isn't my Prepare interceptor being executed

This page last changed on Mar 30, 2006 by scud.

Make sure that the Prepare interceptor comes first before validation interceptor in the stack.

```
<interceptor-stack name="myInterceptorStack">
        ...
     <interceptor-ref name="prepare" />
        ...
     <interceptor-ref name="validation" />
        ...
  </interceptor-stack>
```

Why won't the 'if' tag evaluate a one char string

This page last changed on Mar 30, 2006 by scud.

```
<ww:if test="#myObj.myString == 'A'">
Why doesn't this work when myString is equal to A?
</ww:if>
```

OGNL will interpret 'A' as a char type and not a string. Simple solution - flip the double and single quotes.

```
<ww:if test='#myObj.myString == "A"'>
This works!
</ww:if>
```

Alternatively, you can escape the double quotes in the String:

```
<ww:if test="#myObj.myString == \"A\"">
This works!
</ww:if>
```

FreeMarker

This page last changed on May 19, 2006 by scud.

FreeMarker是一个Java模版语言，它是 JSP 的绝佳替代方案. FreeMarker在你的action result可能需要在Servlet容器环境以外被载入的情况下是理想选择. 例如,如果你希望在你的应用程序中支持plugins,你可能会乐意使用Freemarker，因为那样的话plugins可以支持将所有的action class和view都打包到一个从classloader进行装载的jar文件里面.
关于FreeMarker的更多信息，请访问FreeMarker网站.

FreeMarker与Velocity非常相似, 它们都是可以在Servlet容器外使用的模版语言. WebWork小组更推荐FreeMarker, 而不是Velocity, 这是因为FreeMarker提供了更好的错误报告, 支持JSP标签, 稍多的功能. 当然, 这两种技术都是代替JSP的很好方案.

快速上手

确认配置好你的项目的classpath中的所有依赖以后, 开始使用FreeMarker就非常简单了. 典型情况下只需要 **freemarker.jar**. 除它以外, webwork-default.xml已经配置好了将FreeMarker Result映射到你的模版文件. 你现在可以试验一下如下 **xwork.xml** 配置:

```
<action name="test" class="com.acme.TestAction">
    <result name="success" type="freemarker">test-success.ftl</result>
</action
```

然后写好 **test-success.ftl**:

```
<html>
<head>
    <title>Hello</title>
</head>
<body>

Hello, ${name}

</body>
</html>
```

这里 **name** 是你的action中的一个属性. 这样就可以了! 该文档的余下部分将介绍模版如何被加载, 变量如何解析, tags(标签)也可以使用.

模版加载

Webwork在两个位置查找FreeMarker模版(按顺序):

1. Web应用程序目录(Web application)
2. Class path
   这个顺序对于在完全编译的jar中提供模版很理想, 但是也同时支持在Web应用程序目录中定义这些模版来覆盖jar中的模版文件. 事实上, 这就是为什么你可以覆盖WebWork中默认的UI tags和Form Tags的原理.
   还有, 你可以通过templatePath 上下文变量(context variable)指定一个路径(你的文件系统中的一个目录). 如果指定了该变量, 那么这个目录中的内容将会被优先寻找.

变量解析/决定(Resolution)

在FreeMarker中，变量将会在多个位置进行寻找，顺序如下：

1. 值栈(value stack)
2. action上下文(action context)
3. Request范围(scope)
4. Session范围(scope)
5. Application范围(scope)
6. 内建变量
   注意action上下文在value stack后进行搜索. 这意味着你可以引用变量而不必使用标准的符号(#)前缀，不像在JSP中使用ww:property中必须使用的那种语法(译者注:现在在JSP中也可以不用#而访问默认的ValueStack). 这是一个很好的便利特性，但是小心，它有时也会把你陷进去.

```
<@ww.url id="url" value="http://www.yahoo.com"/>
Click <a xhref="${url}">here</a>!
```

Webwork-FreeMarker整合提供的内建变量如下：

Name	Description
stack	值栈本身，方便使用 ${stack.findString('ognl expr')}的方式调用
action	最近执行的action
response	HttpServletResponse
res	与response相同
request	HttpServletRequest
req	与reqeust相同
session	HttpSession
application	ServletContext
base	request的上下文路径(context path)

标签支持

FreeMarker是很棒的模版语言，因为它完整的支持标签(tag). 参照WebWork提供的 FreeMarker Tags 文档中的如何使用通用(generic) Tags 部分获取更多信息. 除了那些，你还可以使用任何的JSP标签(tag)，就像这样：

```
<#assign mytag=JspTaglibs["/WEB-INF/mytag.tld"]>
<@mytag.tagx attribute1="some ${value}"/>
```

这里 `mytag.tld` 是你使用的JSP标签库的定义文件. 注意：为了使用FreeMarker的这个支持，你必须开启 web.xml 2.1.x compatibility 文档中的 JSPSupportServlet.

提示和技巧

下面是在使用FreeMarker构建WebWork应用程序时的一些有用的进阶功能.

类型转换与本地化

FreeMarker内置支持日期与数字的格式化. 格式化的规则基于action request的地区信息(locale), locale是通过webwork.properties配置的, 它也可以通过I18n Interceptor进行覆盖. 这种方式一般会完美的满足你的需求, 但是你要记住, 这些格式化信息是通过FreeMarker处理的,而不是通过WebWork的类型转换支持实现.
如果你希望WebWork根据你所指定的类型转换处理格式化, 你不应该使用平常的&{...}语法. 取而代之, 你应该使用property标签. 区别在于property标签特别为OGNL表达式设计, 计算它的值, 然后将结果用你指定的Type Conversion转换为String. 平常使用的${...}语法则会使用FreeMarker的表达式语言(EL), 计算它的值, 然后通过内建的格式化规则转化为String. 这些区别甚微, 但是一定要了解.

扩展

有时你可能需要扩展WebWork提供的FreeMarker支持. 最常见的原因是你希望引入你自己的标签, 就像你扩展WebWork内建标签一样.
如果需扩展, 首先要新建一个继承 com.opensymphony.webwork.views.freemarker.FreemarkerManager 并且重载了相应方法的类. 然后将下面代码添加到webwork.properties:

```
webwork.freemarker.manager.classname = com.yourcompany.YourFreeMarkerManager
```

ObjectWrapper设置

如果你熟悉了FreeMarker, 你会发现它的敏感性会带来一些困扰. 最常见的方法就是尝试使用FreeMarker提供的BeanWrapper. 如果你不知道那是什么,别担心. 只要知道这些酒可以了:

WebWorkBeanWrapper继承自默认的FreeMarker BeansWrapper, 提供了基本完全一致的功能, 只是修改了maps处理机制. 一般, FreeMarker有两种操作模式: 一种支持友好的内置的map (?key, ?values, etc),但是只支持String作为key; 或者特殊的内置支持(例如: ?key 返回map的相应方法而不是key), 但是它支持String和String相似的非String作为key. WebWork提供了两种情况下的可选的实现方案.
这种特殊的做法也许会让你迷惑或产生问题. 所以, 你可以将 webwork.properties 中的 *webwork.freemarker.wrapper.altMap*设置为false, 允许替换为常规的BeansWrapper逻辑.

语法注释

如果是FreeMarker 2.3.4, 还支持另外的语法. 这种可选的语法在你感觉你使用的IDE(尤其是IntelliJ IDEA)在默认的语法下运行困难时非常有用. 关于这种语法的更多内容, 请阅读这里

IDEA Plugin

This page last changed on Mar 30, 2006 by scud.

This document is an MRD (Marketing Requirements Document) for the ideal vision of what the WebWork IDEA Plugin will be. At this point, no WebWork IDEA plugin has been created. When it is created, it will be based on EclipseWork.

A key success for WebWork is its ability to provide tools that make developers lives easier. This isn't just some configuration editor, but rather a complex integrated environment where developers can write their code, build JSPs, and have much of the redundant work related to WebWork automated or assisted. WebWork's IDEA integration enables developers to not only write WebWork-based applications faster, but also with much higher quality.

The IDEA Plugin has the following features:

- **Configuration browser** - you can browse namespaces and see both actions and views as well as their relationships. The browser lets you avoid hunting through xwork.xml and all the included files and instead presents a logical view of all your actions and their associated views. The browser also allows for very quick access to an action if you know it by name (and optional namespace).
- **Find by usages** - finding usages on a field will show you were all the views that use that field are located.
- **Refactoring support** - similar to find by usages, it is very useful to rename a field in an action (or even an object that is someone in the stack/object graph) and know that all the views that reference it have been updated as well
- **Internationalization support** - you can select text in a JSP or other views and automatically extract it in to the a choice of the correct resource bundles (package.properties, Action.properties, etc). Conversely, you can press Ctrl-Q on <ww:text/> tags to see what the actual text for a particular i18n key are.
- **Error reporting** - the plugin is excellent at notifying you of errors before you deploy your application. Since many aspects of WebWork are based around loosely coupled and dynamic interactions, the plugin helps you see when you've had a typo such as "doccument" instead of "document". It highlights errors in OGNL expressions, i18n locations, and configuration mixups automatically for you. When these errors happen the plugin also gives you a choice of possible fixes (Ctrl-Enter).
- **Code completion** - when using WebWork tags such as <ww:select value="..."/> pressing Ctrl-Space pops up a code completion dialog that introspects through all the available properties that are in the value stack, including those in your action.
- **OGNL expression evaluator** - When you're editing a page you should be able to have an OGNL expression evaluator that can execute your action and use its properties to evaluate OGNL expressions to show you what you get.

Interceptors

This page last changed on May 19, 2006 by scud.

关于拦截器配置的基本信息

概况

拦截器是动态拦截Action调用的对象.它提供了一种机制可以使开发者可以定义在一个action执行的前后执行的代码,也可以在一个action执行前阻止其执行.同时也是提供了一种可以提取action中可重用的部分的方式.

拦截器必须是无状态的,不能保证为每一个请求或者action创建一个实例.拦截器可以选择短路一个action调用,然后返回一个结果码(如com.opensymphony.xwork.Action#SUCCESS);也可以选择在ActionInvocation#invoke()之前或者之后做一些处理。

Webwork & XWork Interceptors

拦截器以key-value对的方式定义在xwork配置文件中.下面是定义在webwork-default.xml的拦截器.如果您扩展webwork-default包,您就可以使用下面的拦截器.否则您就必须在自己的包中的<interceptors>标签中定义name-class对.

Interceptor	Name	Description
Alias Interceptor	alias	不同的request中的相似参数别名转换.
Chaining Interceptor	chain	使前一个action中的属性在当前action中可用.一般和<result type="chain">一起使用 (在前一个action中).
Component Interceptor	component	是Action中可以使用组件.和components.xml有关
Conversion Error Interceptor	conversionError	把转型错误信息从ActionContext加到Action的field错误
Create Session Interceptor	createSession	自动创建一个HttpSession对象，对于某些需要有HttpSession对象才能正常工作的拦截器(如TokenInterceptor)有用
Execute and Wait Interceptor	execAndWait	后台执行action,发送给用户等待画面.
Exception Interceptor	exception	把异常映射为结果.
File Upload Interceptor	fileUpload	支持文件上传的拦截器.更多信息参见javadoc
I18n Interceptor	i18n	把所选的地域放入用户session
Logger Interceptor	logger	输出Action的名字
Model Driven Interceptor	model-driven	如果action实现了ModelDriven接口,把getModel()的结果堆入valuestack.
Parameters Interceptor	params	把request中的参数传入action.
Prepare Interceptor	prepare	如果action实现了Preparable接口,调用其prepare()方法.

Scope Interceptor	scope	把action的状态存在session或application范围内的简单方法
Servlet Config Interceptor	servlet-config	可以直接访问HttpServletRequest和HttpServletResponse (谨慎使用,这样会使action与Servlet过于紧密)
Static Parameters Interceptor	static-params	把定义在xwork.xml中的<action>标签下的<param>标签中的参数传入action
Timer Interceptor	timer	输出action执行时间(包括内嵌拦截器和视图)
Token Interceptor	token	检查传到action中的token,防止多次提交
Token Session Interceptor	token-session	功能同上,token储存在session中
Validation Interceptor	validation	执行定义在xxxAction-validation.xml中的校验器
Workflow Interceptor	workflow	调用action类中的validate方法,如果产生错误返回INPUT画面.
Parameter Filter Interceptor	N/A	根据合法的方法列表来去除一些参数

方法过滤

抽象的拦截器可以通过指定included/excluded方法列表来实现可选择性

可以设置的参数如下:

- excludeMethods - 被排除的方法
- includeMethods - 被包含的方法

注意: 如果一个方法的名字同时出现在includeMethods和includeMethods里，它会被当作包含的方法。也就是，includeMethods优先于excludeMethods.

扩展了这一能力的拦截器有:

- TokenInterceptor
- TokenSessionStoreInterceptor
- DefaultWorkflowInterceptor
- ValidationInterceptor

拦截器参数覆盖

拦截器的参数可以通过如下方式被覆盖

方法1:

```
<action name="myAction" class="myActionClass">
  <interceptor-ref name="exception"/>
    <interceptor-ref name="alias"/>
    <interceptor-ref name="params"/>
    <interceptor-ref name="servlet-config"/>
```

```
        <interceptor-ref name="prepare"/>
        <interceptor-ref name="i18n"/>
        <interceptor-ref name="chain"/>
        <interceptor-ref name="model-driven"/>
        <interceptor-ref name="fileUpload"/>
        <interceptor-ref name="static-params"/>
        <interceptor-ref name="params"/>
        <interceptor-ref name="conversionError"/>
        <interceptor-ref name="validation">
          <param name="excludeMethods">myValidationExcudeMethod</param>
        </interceptor-ref>
        <interceptor-ref name="workflow">
          <param name="excludeMethods">myWorkflowExcludeMethod</param>
        </interceptor-ref>
    </action>
```

方法2:

```
    <action name="myAction" class="myActionClass">
      <interceptor-ref name="defaultStack">
        <param name="validator.excludeMethods">myValidationExcludeMethod</param>
        <param name="workflow.excludeMethods">myWorkflowExcludeMethod</param>
      </interceptor-ref>
    </action>
```

在第一个方法中，整个默认栈都被复制，然后根据需要改变参数。

在第二个方法中，我们引用了已经存在的拦截器栈。在这个例子中这个栈是default-stack，然后覆盖了validator和workflow拦截器的excludeMethods参数。注意在这个标签的name属性中有一个点(.)，点之前的单词表示要被覆盖参数的拦截器名，点之后的表示参数。形式如下:

```
<拦截器名>.<参数名>
```

也要**注意**到,在这个例子中如果name属性用来表示一个拦截器栈,就像指向一个拦截器本身,那只能用上面描述的第一种方法.

拦截器执行顺序

拦截器提供了极好的方式去包装 前/后 处理.这种概念减少了代码重复(就像AOP).

```
    <interceptor-stack name="xaStack">
      <interceptor-ref name="thisWillRunFirstInterceptor"/>
      <interceptor-ref name="thisWillRunNextInterceptor"/>
      <interceptor-ref name="followedByThisInterceptor"/>
      <interceptor-ref name="thisWillRunLastInterceptor"/>
    </interceptor-stack>
```

注意一些拦截器会打乱stack/chain/flow...所以顺序非常重要.

实现了com.opensymphony.xwork.interceptor.PreResultListener的拦截器在Action之后Result之前执行.

```
    executesthisWillRunFirstInterceptor
      thisWillRunNextInterceptor
        followedByThisInterceptor
          thisWillRunLastInterceptor
        MyAction1
        MyAction2 (chain)
        MyPreResultListener
        MyResult (result)
      thisWillRunLastInterceptor
    followedByThisInterceptor
  thisWillRunNextInterceptor
thisWillRunFirstInterceptor
```

Alias Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器的目的是将一个参数名映射为另一个参数名.它可以作为在不同action之间共享不同名字的相同参数的粘合剂,这一点对于实现action链很有用.

Action的别名表达式应该是这样的形式:

```
#{ "name1" : "alias1", "name2" : "alias2" }
```

. 这个意思是说,如果一个action(或者stack中的其他东西)有一个叫name1的属性,这个action应用了这个拦截器,且有alias1的setter方法, name1的值会被设置到 alias1上.

参数

- aliasesKey (可选的) - action中的别名Map的名字 (默认为aliases).

扩展这个拦截器

这个拦截器没有可扩展的地方.

例子

```
<action name="someAction" class="com.examples.SomeAction">
  <\!-\- The value for the foo parameter will be applied as if it were named bar \-->
  <param name="aliases">#{ 'foo' : 'bar' } </param>
  <\!-\- note: the alias interceptor is included with the defaultStack in webwork-default.xml
\-->
  <interceptor-ref name="alias"/>
  <interceptor-ref name="basicStack"/>
  <result name="success">good_result.ftl</result>
</action>
```

Chaining Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器可以把ValueStack中除实现了Unchainable接口以外的所有对象全都复制到当前正在执行的Action。拦截器的可选参数includes和excludes可以用来控制哪些参数可以被复制以及这些参数怎样被复制。includes 或 excludes 只能同时指定一个。如果两个都指定效果就等于没有指定。更多信息查看javadoc中的{@link OgnlUtil#copy(Object, Object, java.util.Map, java.util.Collection, java.util.Collection)}

。
需要注意的一点是如果ValueStack中没有任何对象，这个拦截器不做任何事。这就意味两点：一、你可以把它应用于任何Action而不必担心产生负面影响。二、你必须在调用一个actioin之前确保在stack中有要传递的值。最常见的使用方式是和 chain 结果类型一起使用，这样就可以实现action chaining的效果了。

参数

- excludes (optional) - 不需要复制的参数，其他的都复制
- includes (optional) - 一定要复制的参数，其他的都不复制

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="basicStack"/>
    <result name="success" type="chain">otherAction</result>
</action>

<action name="otherAction" class="com.examples.OtherAction">
    <interceptor-ref name="chain"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Component Interceptor

This page last changed on May 19, 2006 by scud.

一个简单的拦截器可以把WebWork的IOC容器ComponentManager应用于执行Action。注意，WebWork的IOC已经废弃掉了，推荐使用其他的解决方案，如Spring。

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="componentStack"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Conversion Error Interceptor

This page last changed on May 19, 2006 by scud.

要全面地了解这个拦截器，最好查看一下这个拦截器的子类(译者注:原文中是subclass，但是应该是父类
)ConversionErrorInterceptor的JavaDocs:

> 这个拦截器把在ActionContext的conversionErrors Map中找到的所有错误添加为fieldError(action需要实现ValidationAware接口)。同时把有验证错误的field的原始值被保存下来，这样任何后面对于这个值的请求得到的都是原始值而不是action中的值。这一点很重要，因为如果值"abc"被提交了，但是它不能被转型为int，我们希望显示回来的是原来的"abc",而不是一个int值(像0,这样的值不易被用户察觉是输入错误了)

再来看看拦截器WebWorkConversionErrorInterceptor自己的JavaDocs:

> 它继承于ConversionErrorInterceptor,但是仅仅是在field值不是Null，或者不是""，或者不是{""}(只有一个元素的字符串数组，并且这个元素还是个空字符串)的情况下把转型错误从ActionContext加到Action的fieldErrors中。详细信息查看ConversionErrorInterceptor和类型转换文档.

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
 <interceptor-ref name="params"/>
 <interceptor-ref name="conversionError"/>
 <result name="success">good_result.ftl</result>
 </action>
```

Create Session Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器创建HttpSession对象.
在用到FreeMarker模板的<@ww.tokten>标签时很有用。这个标签需要在freemarker把response提交给客户端时必须已经存在一个HttpSession对象。

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="create-session"/>
    <interceptor-ref name="defaultStack"/>
    <result name="input">input_with_token_tag.ftl</result>
</action>
```

Exception Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器提供了异常处理的主要核心功能.异常处理允许你把一个异常映射到一个结果码,就像是action返回一个结果码,而不是抛出意想不到的异常.当一个异常出现,它会被包装成ExceptionHolder放到栈中,这样我们就可以很容易的在结果中使用了

注意:你可以在配置文件中任何一点配置异常映射,但是如果你没有把这个拦截器放在你的拦截器栈中,就不会产生任何影响.建议把这个拦截器放置在栈的第一个,这样可以保证它能捕捉到所有action甚至其它拦截器中出现的异常.

参数

无

扩展

如果你想自定义异常的发布方式,你可以复写{@link #publishException(com.opensymphony.xwork.ActionInvocation, ExceptionHolder)}.默认实现是把得到的ExceptionHolder放到值栈中.你可以根据需要添加其它逻辑.

例子

```
<xwork>
    <include file="webwork-default.xml"/>

    <package name="default" extends="webwork-default">
        <global-results>
            <result name="success" type="freemarker">error.ftl</result>
        </global-results>

        <global-exception-mappings>
            <exception-mapping exception="java.lang.Exception" result="error"/>
        </global-exception-mappings>

        <action name="test">
            <interceptor-ref name="exception"/>
            <interceptor-ref name="basicStack"/>
            <exception-mapping exception="com.acme.CustomException" result="custom_error"/>
            <result name="custom_error">custom_error.ftl</result>
            <result name="success" type="freemarker">test.ftl</result>
        </action>
    </package>
</xwork>
```

Execute and Wait Interceptor

This page last changed on May 19, 2006 by scud.

ExecuteAndWaitInterceptor拦截器能够让一个执行时间较长的Action在后台执行，并向用户显示进度信息。当一个Action的执行时间会超过5或10分钟时，它可以防止HTTP请求超时。

这个拦截器的使用方法很直接。假如你已经引入了webwork-default.xml，这个拦截器就已经配置好了，只是没有出现在默认的拦截器栈中。这个拦截器的特性要求它必须被放在栈的**最后**。

这个拦截器是基于每个Session工作，也就是说同一个Action不能在同一个的Sessioin中运行一次以上。初始请求和后续请求(在Action执行结束以前)将返回**wait** 结果。**wait结果负责让后续的请求返回到这个Action，并显示更新的进度信息。**

如果没有找到"wait"结果,WebWork会自动生成一个wait结果.这个结果是用FreeMarker做的,所以需要Freemarker支持才能正常工作。如果你不想在程序中加入FreeMarker，那就必须自己实现一个wait结果。这一般来说是有必要的，因为默认的wait页面很简单。

无论何时只要返回了wait结果，**正在后台运行的action将会被放到栈顶**。这就允许你在等待页面中显示进度数据，例如计数器。通过让等待页面自动重新加载这个action(它会被这个拦截器短路),你可以实现一个自己更新的进度条.

这个拦截器也支持初始等待延迟.初始等待延迟就是我们可以让服务器在显示等待页面之前延迟一段时间(用毫秒表示).在这段时间里这个拦截器每隔100毫秒检查后台进程是不是过早的执行完了.如果出于某种原因这个任务并没有使用很长时间就完成了,等待画面就不会显示给用户.

这个对于执行时间较长的搜索action很有用.我们可以延迟2000毫秒,如果搜索时间较短就立刻返回结果,如果较长就返回等待画面.

注意:因为这个action将会以单独的线程执行,所以你不能用ActionContext,因为它是ThreadLocal.这也就是说如果你要访问session数据，你必须实现SessionAware结构而不是调用ActionContext.getSesion().

这个拦截器使用的线程会被命名为 **actionNameBrackgroundProcess, 例如searchAction会以一个叫searchBackgroundProcess的线程执行。**

参数

- threadPriority(可选) 指定线程的优先级.默认是Thread.NORM_PRIORITY
- delay(可选) 初始延迟的毫秒,在显示等待画面(返回wait结果码)前的等待时间.默认是没有等待延迟.
- delaySleepInterval(可选) 只能和delay一起用.检查后台进程是否执行完毕的周期(毫秒).默认是100 毫秒

扩展

如果你需要在后台线程调用前后做一些准备或处理,你可以扩展BackgroundProcess类并实现beforeInvocation() 和afterInvocation()方法.在你需要获得和释放后台进程要成功执行所需的资源时,这一点很有用. 要使用你扩展了的后台进程类,你需要扩展ExecuteAndWaitInterceptor并实现getNewBackgroundProcess()方法.

例子

```
<action name="someAction" class="com.examples.SomeAction">
     <interceptor-ref name="completeStack"/>
     <interceptor-ref name="execAndWait"/>
     <result name="wait">longRunningAction-wait.jsp</result>
     <result name="success">longRunningAction-success.jsp</result>
 </action>
```

```
<%@ taglib prefix="ww" uri="/webwork" %>
 <html>
   <head>
     <title>Please wait</title>
     <meta http-equiv="refresh" content="5;url=<ww:url includeParams="all" />"/>
   </head>
   <body>
     Please wait while we process your request.
     Click <a href="<ww:url includeParams="all" />"></a> if this page does not reload
automatically.
   </body>
 </html>
```

例子代码2:
这个例子在等待画面显示给用户前等待2分钟(2000毫秒).这样如果这个需要长时间的处理没有花费很长时间,用户就不会看到等待画面.

```
<action name="someAction" class="com.examples.SomeAction">
     <interceptor-ref name="completeStack"/>
     <interceptor-ref name="execAndWait">
         <param name="delay">2000<param>
     <interceptor-ref>
     <result name="wait">longRunningAction-wait.jsp</result>
     <result name="success">longRunningAction-success.jsp</result>
</action>
```

例子代码3:
这个例子在等待画面显示给用户前等待1分钟(1000毫秒).并且每隔50毫秒检查一下后台进程有没有执行完毕,如果完成了它就立刻返回,不用等到1分钟,用户不会看到等待画面.

```
<action name="someAction" class="com.examples.SomeAction">
     <interceptor-ref name="completeStack"/>
     <interceptor-ref name="execAndWait">
         <param name="delay">1000<param>
         <param name="delaySleepInterval">50<param>
     <interceptor-ref>
     <result name="wait">longRunningAction-wait.jsp</result>
     <result name="success">longRunningAction-success.jsp</result>
 </action>
```

HibernateAndSpringEnabledExecuteAndWaitInterceptor

This page last changed on May 19, 2006 by scud.

下面是一个扩展ExecuteAndWaitInterceptor的例子.

这段代码的目的是允许在后台执行的同时可以访问到相同的打开的Hibernate Session对象

SessionFactory的依赖是通过Spring注射到OpenSessionExecuteAndWaitInterceptor里面的. 你也可以使用其它的依赖注入方法, 如果更适合的话. 通过覆写getNewBackgroundProcess()方法, 这个拦截器使用自定义OpenSessionBackgroundProcess而不是WebWork默认的.

覆写了OpenSessionBackgroundProcess中的beforeInvocation()和afterInvocation()方法来确保在后台处理过程中session一直处于打开状态, 任何Spring的事务管理都是可用的.

这个代码依赖于Spring和Hibernate, 所以你不可能在Webwork的发布包中看见它. 尽管如此, 这是个扩展misc:Execute and Wait Interceptor的一个很有用的例子.

```
import net.sf.hibernate.SessionFactory;

import com.opensymphony.webwork.interceptor.BackgroundProcess;
import com.opensymphony.webwork.interceptor.ExecuteAndWaitInterceptor;
import com.opensymphony.xwork.ActionInvocation;


/**
 * The OpenSessionExecuteAndWaitInterceptor will obtain a Hibernate
 * Session Factory from a Spring.
 *
 * The session factory will then be passed to the BackgroundProcess,
 * to open a session, enable Spring's transaction management
 * capabilities, and bind the Session to the background thread.
 *
 */
public class OpenSessionExecuteAndWaitInterceptor extends ExecuteAndWaitInterceptor {

    SessionFactory sessionFactory;


        public SessionFactory getSessionFactory() {
                return sessionFactory;
        }


        public void setSessionFactory(SessionFactory sessionFactory) {
                this.sessionFactory = sessionFactory;
        }

        protected BackgroundProcess getNewBackgroundProcess(String arg0, ActionInvocation arg1,
int arg2) {
                return new OpenSessionBackgroundProcess(arg0, arg1, arg2, sessionFactory);
        }

}
```

```
import net.sf.hibernate.FlushMode;
import net.sf.hibernate.Session;
import net.sf.hibernate.SessionFactory;

import org.springframework.orm.hibernate.SessionFactoryUtils;
import org.springframework.orm.hibernate.SessionHolder;
import org.springframework.transaction.support.TransactionSynchronizationManager;

import com.opensymphony.webwork.interceptor.BackgroundProcess;
import com.opensymphony.xwork.ActionInvocation;
```

```
/**
 * The OpenSessionBackgroundProcess, when instantiated with a
 * HibernateSessionFactory, will open a session, enable Spring's transaction
 * management capabilities, and bind the Session to the background thread.
 *
 */
public class OpenSessionBackgroundProcess extends BackgroundProcess {

        SessionFactory sessionFactory;

        Session openSession;

        public OpenSessionBackgroundProcess(String name,
                        ActionInvocation invocation, int threadPriority,
                        SessionFactory factory) {
                super(name, invocation, threadPriority);
                this.sessionFactory = factory;
        }

        protected void beforeInvocation() throws Exception {
                openSession = SessionFactoryUtils.getSession(sessionFactory, true);
                openSession.setFlushMode(FlushMode.NEVER);
                TransactionSynchronizationManager.bindResource(sessionFactory,
                                new SessionHolder(openSession));
                super.beforeInvocation();
        }

        protected void afterInvocation() throws Exception {
                super.afterInvocation();
                TransactionSynchronizationManager.unbindResource(sessionFactory);
                SessionFactoryUtils
                                .closeSessionIfNecessary(openSession, sessionFactory);
        }


}
```

File Upload Interceptor

This page last changed on Apr 16, 2006 by scud.

这个拦截器基于MultiPartRequestWrapper,它可以应用于任何一个包含文件的请求. 它提供以下参数, 其中的[File Name]是被HTML表单中上传的文件的名字

- [File Name] : File - 实际的文件
- [File Name]ContentType : String - 文件的ContentType
- [File Name]FileName : String - 上传文件的实际名字(不是HMTL标签的名字)

你只要在你的action中提供任何一种上面形式的set方法就可以访问到这些文件.例如setDocument(得到Document文件),setDocumentContentType(String类型的contentType),等等.

如果你的action实现了ValidationAware接口,这个拦截器就可以添加几种字段错误.这些错误信息是基于存储在webwork-messages.properties文件中的一些i18n值,这个文件是所有i18n请求的默认文件.你可以在自己消息文件的复写以下key的消息文字:

- webwork.messages.error.uploading - 文件不能上传的通用错误信息
- webwork.messages.error.file.too.large - 上传文件长度过大的错误信息
- webwork.messages.error.content.type.not.allowed - 当上传文件不符合指定的contentType

参数

- maximumSize (可选) - 这个拦截器允许的上传到action中的文件最大长度(以byte为单位). 注意这个参数和在webwork.properties中定义的属性没有关系,默认2MB
- allowedTypes (可选) - 以逗号分割的contentType类型列表(例如text/html),这些列表是这个拦截器允许的可以传到action中的contentType.如果没有指定就是允许任何上传类型.

扩展

你可以扩展这个拦截器并复写acceptFile方法来实现更好的控制什么文件是被支持的什么是不被支持.

例子

```
<action name="doUpload" class="com.examples.UploadAction">
    <interceptor-ref name="fileUpload"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

然后你需要设置上传文件中的表单中的encoding属性为multipart/form-data

```
<ww:form action="doUpload" method="post" enctype="multipart/form-data">
      <ww:file name="upload" label="File"/>
      <ww:submit/>
</ww:form>
```

然后在action中代码中加入set方法,你可以访问到File对象了.

```
public com.examples.UploadAction implemements Action {
      private File file;
      private String contentType;
      private String filename;

      public void setUpload(File file) {
         this.file = file;
      }

      public void setUploadContentType(String contentType) {
         this.contentType = contentType;
      }

      public void setUploadFileName(String filename) {
         this.filename = filename;
      }

      ...
```

设置参数的例子

```
<interceptor-ref name="fileUpload">
  <param name="allowedTypes">
     image/png,image/gif,image/jpeg
  </param>
</interceptor-ref>
```

I18n Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器负责把session中指定的区域设置为当前请求的区域设置.并且这个拦截器还会根据请求中的参数来改变区域设置.这就说你可以用来动态的改变用户Session中区域设置.这对于实现多语言支持并允许用户在任何时候更改语言很有用.区域设置参数在这个拦截器执行的时候会被删除,以确保action中相应的属性不会被设置(例如request_locale).

例如,使用默认的参数,一个像foo.action?request_locale=en_US这样的请求就可以把US English作为区域设置保存到用户的Session中,以便被将来的请求所使用.

参数

- parameterName (可选) - HTTP请求参数的名字,表示要切换和保存到Session的区域设置.默认是request_locale
- attributeName (可选) - Session中用来存放区域设置的Key的名字,默认是WW_TRANS_I18N_LOCALE

扩展

无

Examples

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="i18n"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Logger Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器记录一个action执行的开始和结束.(只用英语,不是国际化的)

参数

无

扩展

无

例子

```
<!-- prints out a message before and after the immediate action execution -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="completeStack"/>
    <interceptor-ref name="logger"/>
    <result name="success">good_result.ftl</result>
</action>

<!-- prints out a message before any more interceptors continue and after they have finished
-->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="logger"/>
    <interceptor-ref name="completeStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Model Driven Interceptor

This page last changed on May 19, 2006 by scud.

监视模型驱动的Action,并把action中的模型放到值栈中.

注意: 如果想让参数被设置到模型中,必须把ModelDrivenInterceptor放在StaticParametersInterceptor和ParametersInterceptor两个拦截器的前面

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="model-driven"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Parameter Filter Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器可以过滤掉剩下的栈或你的action中的参数.你可以在一个action中使用多个Parameter Filter Interceptor.例如你可以在默认栈中加入一个,这样过滤掉每一个action中都不要的参数。也可以为单独的action中添加一个来过滤这个action参数.

参数

- allowed - 逗号分割的允许传入action参数前缀列表
- blocked - 逗号分割的不允许传入action的参数前缀列表
- defaultBlock - boolean类型(默认false) 是否在默认的情况下阻拦指定参数.如果true,那么只有在允许列表里参数才可以传入action.

最简单的配置时的参数过滤方式是,如果一个字符串在允许或阻拦列表里,那么任何这个字符串对应的对象中的所有成员属性将被允许或阻拦.

例如, 如果参数是如下:

- blocked: person,person.address.createDate,personDao
- allowed: person.address
- defaultBlock: false
  参数 person.name, person.phoneNum 等会被阻拦,因为'person'在阻拦列表里。 但是person.address.street和person.address.city会被允许，因为'person.address'在允许列表里面。

Parameters Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器把所有参数从ActionContext的getParameters()方法取出,然后通过OgnlValueStack的setValue(String, Object)方法放到值栈中.通常是把请求表单中的数据放到action要用的值栈中.注意,参数表中必须有一个String类型的key和String[]类型的值.

由于参数的名字可以作为ONGL表达式被执行,所以这个拦截器中加入了一些安全机制. 如果一个参数名的表达式中存在赋值(=),多重表达式(,),或者指向环境中的任何对象的引用,拦截器就不会处理参数表中的任何值.这是通过acceptableName(String)方法来实现的.出此之外如果action实现了ParameterNameAware接口,这个拦截器还会检查action某个参数是否应该被设置.

处了这些限制之外,一个名为XWorkMethodAccessor#DENY_METHOD_EXECUTION的标识被设置,这样任何方法将不允许被执行.这就是说任何诸如person.doSomething()或者person.getName()这样的表达式都会被严格的禁止.这可以保证你的程序不会被恶意的用户攻击.

当这个拦截器被调用时,一个名为InstantiatingNullHandler#CREATE_NULL_OBJECTS的标识被设置来保证在可能的情况下null引用被自动创建.更多信息可以参见类型转换文档和InstantiatingNullHandler的javadocs.

最后,第三个名为XWorkConverter#REPORT_CONVERSION_ERRORS的标识被设置来指明每一个在把值转换成最终类型(String[] -> int)时产生的错误为不可恢复的错误. 这个标识设置以后任何类型转换错误会被报告到action context中.更多信息可以参见类型转换文档和XWorkConverter的javadocs.

如果你想查看参数的详细信息,可以把拦截器的logging设置为DEBUG级别. 这样所有参数的键和值会为详细的报告出来.

关于参数名字限制的方式可以查看ParameterNameAware的javadocs:

想要声明可以接受的参数的action应该实现这个接口. 它通常和ParametersInterceptor一起工作. 例如,一些action想要创建一个可被接受参数列表或者拒绝接受到参数列表, 来防止客户端设置其它的不希望的(或者可能是有危险的) 参数.

参数

无

扩展

扩展这个拦截器的最好方式是让你的action实现ParameterNameAware接口. 当然,如果你希望是应用于全局而不是仅仅应用于某一个action,你可以扩展这个拦截器并重写acceptableName(String)方法.

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="params"/>
    <result name="success">good_result.ftl</result>
</action>
```

Prepare Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器执行实现了Preparable接口的action的prepare()方法. 当你需要在Action中的方法执行之前执行一些逻辑,这个拦截器就很有用了.

比较典型的应用是执行一些逻辑从数据库中加载一个对象,这样当参数需要被设置时就可以根据这个对象来设置. 例如, 假设你有一个User对象,它有两个属性: id和name. 如果params拦截器执行两次(在这个拦截器之前一次,之后一次), 你就可以通过id属性加载User这个对象, 然后当params拦截器第二次执行时,user.name这个参数将会根据前面从数据库读出来的User对象来设置. 更多信息见下面的例子.

参数

无

扩展

无

例子

```
<!-- Calls the params interceptor twice, allowing you to pre-load data for the second time
parameters are set -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="params"/>
    <interceptor-ref name="prepare"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Scope Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器是为了实现在WebWork中实现类似向导的功能而设计的. 其中一个问题就是,一些应用程序有一些在application范围内的公用参数, 例如pageLen(每个页面的记录数). 有了这个拦截器就不用让每个Action都检查这样的参数是否存在,而是让这个拦截器去检查并把这些参数从Session中取出来.

在Action开始的时候,这个拦截器会把Session/Application中的以action类名,action名或者任何指定的键为主键的值取出设置为一列属性. 当Action执行结束这列属性被取回放到session或者application的上下文环境中.

为了保证每个action的执行是一致的,这个拦截器采用了session级的锁定. 这种方式可以确保每一个action的执行在session级上都是原子的. 它不确保application级别上的一致性, 尽管有理由这么做. Application级别上的一致性会导致很大的性能问题.

注意这个拦截器会在结果呈现以前(使用一个PreResultListener)取得一个action的属性的快照, 而不是在action执行之后. 这是有原因的: 在这个时刻我们知道action的状态是"complete",它的值可能依赖于其它剩下的栈,特别是依赖于嵌套的拦截器的值.

参数

- session - 一个限制在session范围内的action属性列表
- application - 一个限制在application范围内的属性列表
- key - 一个session/application属性的主键前缀, 可以包括如下值:
  - CLASS - 根据action的名称空间和类来创建一个不重复的主键前缀, 这是默认值
  - ACTION - 根据action的名称空间和action的名字创建一个不重复的前缀
  - 任意一个字符串作为主键前缀
- type - 下列值之一
  - start - 表示这是一个向导的开始action, 所有session范围内的属性将被重设为默认值
  - end - 表示在这个action执行以后,session范围内的属性将会从session中清除
  - any 其它值或者没有值意思是这是一个中间的action,在action执行前把把属性从session中取出放到action中, 执行以后从action取出放回session
- sessionReset - boolean类型,这个值将使所有Session范围内的值重设为Action的默认值或者application范围内的值. 它类似于type="start",而且事实上它们做着同样的事情, 但是在我们的队伍里, 这个值在语义上更准确. 我们以两种模式使用Session范围-有时是类似于向导的action序列,有开始和结束; 有时我们只想简单的重设Session值.

扩展

无

例子

```
<!-- ##filter#orderBy##########action#######,
     ############scope###. #session############Session##########.
     ######Application######-->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="basicStack"/>
    <interceptor-ref name="hibernate"/>
    <interceptor-ref name="scope">
        <param name="session">filter,orderBy</param>
        <param name="autoCreateSession">true</param>
    </interceptor-ref>
```

```
        <result name="success">good_result.ftl</result>
    </action>
```

Servlet Config Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器会根据action实现的接口来设置action的属性. 例如, 如果action实现了ParameterAware接口 那么ActionContext中的参数就会被设置.

这个拦截器是设计用来设置action需要的所有参数的, 如果这个action需要看到servlet参数, servlet上下文, session等. 支持的接口如下:

- ServletContextAware
- ServletRequestAware
- ServletResponseAware
- ParameterAware
- SessionAware
- ApplicationAware
- PrincipalAware

参数

无

扩展

无

Examples

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="servlet-config"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Static Parameters Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器把定义在action配置中的静态参数设置到action中. 如果action实现了Parameterizable一个静态参数的Map也会直接传给action.

参数一般在xwork.xml中用<param>参数定义.

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="static-params">
         <param name="parse">true</param>
    </interceptor-ref>
    <result name="success">good_result.ftl</result>
</action>
```

Timer Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器以毫秒记录action执行时间. 为了使这个拦截器正常工作, logging框架必须至少是INFO级别. 这个拦截器依赖于http://jakarta.apache.org/commons/logging/ 来报告执行时间.

参数

TODO: Describe the paramters for this Interceptor.

扩展

TODO: Discuss some possible extension of the Interceptor.

例子

```
<!-- ###action#### -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="completeStack"/>
    <interceptor-ref name="timer"/>
    <result name="success">good_result.ftl</result>
</action>

<!-- ##action##########-->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="timer"/>
    <interceptor-ref name="completeStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Token Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器可以保证一个令牌对应一个请求。确保后退按钮和两次提交不会产生不希望的效果。 例如你可以使用这个来防止粗心的用户在在线商店点了两下"结帐"按钮。这个拦截器使用了非常简单的机制来处理非法令牌：返回一个invliad.token的结果，这样你就可以在action配置中做映射了。一个复杂一些的实现是TokenSessionStoreInterceptor，可以在发现非法令牌时提供更好的处理逻辑。

注意：为了设置表单的令牌，你必须使用**token**标签。 这个标签放在表单中，并且这个表单是提交到受这个拦截器保护的action:任何不提供令牌(使用token标签产生的)的请求将被处理为非法请求

国际化注意事项：这个拦截器用下面的键作为错误信息。

注意：因为这个拦截器是扩展于MethodFilterInterceptor，所以可以决定在action中的哪些方法上应用它。更多信息参见MethodFilterInterceptor

参数

无

扩展

一般情况下用户不必扩展这个拦截器，TokenSessionStoreInterceptor扩展了这个拦截器。其中的protected方法handleInvalidToken和handleValidToken可以用来处理更有意义的逻辑，就像TokenSessionStoreInterceptor做的那样。

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="token"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>

<-- #######action#myMethod######### -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="token">
          <param name="excludeMethods">myMethod</param>
    </interceptor-ref name="token"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Token Session Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器扩展了TokenInterceptor，提供了更好的逻辑来处理非法令牌。不像普通的令牌拦截器，这个拦截器会通过Session提供灵活的故障切换机制来处理多个请求事件。就是说，它会阻拦后续的请求直道第一个请求完成，这样系统就不会返回invalid.token的结果码，而是试图显示原始的那个合法的请求应该显示的东西，就像没有多个请求在一处重复提交一样。

注意 ：因为这个其扩展了MethodFilterInterceptor，它可以决定是否只把这个功能应用于某些方法上。详细信息参见MethodFilterInterceptor

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="token-session/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>

<-- In this case, myMethod of the action class will not
       get checked for invalidity of token -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="token-session>
        <param name="excludeMethods">myMethod</param>
    </interceptor-ref name="token-session>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
```

Validation Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器通过标准的验证框架运行action。这个标准的输入验证框架根据验证规则(在类似ActionClass-validation.xml这样的文件中定义的)来检查action，然后添加字段级别和action级别的错误信息(前提是你的action必须实现了com.opensymphony.xwork.ValidationAware接口)。这个拦截器一般是栈中的最后一个(或者倒数第二个)。这样可以保证所有的值已经被设置到action中了。

如果要调用的方法名已经在excludeMethods参数中被指定了，这个拦截器不会做任何事情。excludeMethods参数是用逗号隔开的方法名列表。例如，如果你设置excludeMethods参数为"input, back"，那么指向foo!input.action和foo!back.action请求就会被这个拦截器跳过。

注意，这个过程不会对com.opensymphony.xwork.Validateable接口做任何事情，只是简单的向action中添加一些错误信息。action请求的流向不会因为这个拦截器而改变。一般要把这个拦截器和workflow拦截器一起使用。

注意 ：因为这个其扩展了MethodFilterInterceptor, 它可以决定是否只把这个功能应用于某些方法上。详细信息参见MethodFilterInterceptor

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="params"/>
    <interceptor-ref name="validation"/>
    <interceptor-ref name="workflow"/>
    <result name="success">good_result.ftl</result>
</action>

<-- in the following case myMethod of the action class will not
       get validated -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="params"/>
    <interceptor-ref name="validation">
        <param name="excludeMethods">myMethod</param>
    </interceptor-ref>
    <interceptor-ref name="workflow"/>
    <result name="success">good_result.ftl</result>
</action>
```

Workflow Interceptor

This page last changed on May 19, 2006 by scud.

这个拦截器在允许拦截器链继续之前，做一些基础的输入验证流程工作。

如果要调用的方法名已经在excludeMethods参数中被指定了，这个拦截器不会做任何事情。excludeMethods参数是用逗号隔开的方法名列表。例如，如果你设置excludeMethods参数为"input, back"，那么指向foo!input.action和foo!back.action请求就会被这个拦截器跳过。

执行的流程顺序如下

- 如果Action实现了Validateable接口， Action的{@link Validateable#validate() validate} 方法将会被调用
- 接下来，如果Action实现了ValidationAware接口，Action的{@link ValidationAware#hasErrors() hasErrors} 方法会被调用。如果这个方法返回true，这个拦截器中断拦截器链的执行，立即返回Action#INPUT

注意: 如果action没有实现这两个接口中的任何一个，拦截器不做任何处理。这个拦截器一般和输入验证拦截器一起使用。然而这不是必须的，尤其是你在validate()方法中自己实现了验证规则，而不是写在xml文件中的时候。

注意 ：因为这个其扩展了MethodFilterInterceptor，它可以决定是否只把这个功能应用于某些方法上。详细信息参见MethodFilterInterceptor

参数

无

扩展

无

例子

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="params"/>
    <interceptor-ref name="validation"/>
    <interceptor-ref name="workflow"/>
    <result name="success">good_result.ftl</result>
</action>

<-- In this case myMethod of the action class will not pass through
       the workflow process -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="params"/>
    <interceptor-ref name="validation"/>
    <interceptor-ref name="workflow">
        <param name="excludeMethods">myMethod</param>
    </interceptor-ref name="workflow">
    <result name="success">good_result.ftl</result>
</action>
```

Internationalization

This page last changed on May 19, 2006 by scud.

描述

WebWork在两个不同的地方支持国际化(简称 i18n): UI标签和 action/field 错误信息.

- Tags 一般指 i18n 和 text 标签
- 校验

资源包搜索顺序

资源包按照下面的顺序搜索:

1. ActionClass.properties
2. BaseClass.properties (所有的基类直到 Object.properties)
3. Interface.properties (每一个接口和子接口)
4. ModelDriven 的 model (如果实现了 ModelDriven), 对于 model 对象从第一步重复执行
5. package.properties (类所在的目录和每个父目录直到根目录)
6. 搜索 i18n message key 自己的层次关系
7. 全局资源属性 (webwork.custom.i18n.resources) ,在webwork.properties 里定义的

要了解更多的信息,请查阅 LocalizedTextUtil 类.

Package 层次关系

为了把 #5 阐述清楚, 当它在一个包的层次中遍历的时候, WebWork会寻找一个叫package.properties的文件:

```
com/
acme/
package.properties
actions/
package.properties
FooAction.java
FooAction.properties
```

如果 FooAction.properties 不存在,就会寻找com/acme/action/package.properties, 如果没有找到 com/acme/package.properties, 如果没有找到 com/package.properties, 依此类推.

例子

使用 getText()

为了显示 i18n 文字,你可以在property 标签里使用一个对getText()的调用,或者任何其它标签,例如UI标签(这尤其对UI标签的label有用):

```
<ww:property value="getText('some.key')" />
```

Text 标签

你也可以使用 text 标签:

```
<ww:text name="some.key" />
```

I18n 标签

也要注意到有一个i18n标签会把一个资源包推送到stack中,允许你显示不是前面提到的搜索层次关系中的资源包包含的文字.

```
<ww:i18n name="some.package.bundle" >
     <ww:text name="some.key" />
</ww:i18n>
```

在SiteMesh的装饰页面里面使用国际化是可能的,但是存在几个怪异之处.查阅 SiteMesh 页面来了解如何整合WebWork和SiteMesh,包括整合的技巧.

I18n 拦截器

查看 I18n Interceptor 了解更多信息. 它基本上就是在每个请求中把一个locale放到ActionContext Map中. Webwork (组件, ActionSupport 等.) 会感知到这个,并且每个i18n相关的部分都会使用这个locale.这也是一个基于request的改变locale的优雅方式.

在webwork.properties里配置的全局资源包(webwork.custom.i18n.resources)

一个全局的资源包可以通过在webwork.properties里面设定 'webwork.custom.i18n.resources' 属性来做到. locale也可以通过在webwork.properties里面的 'webwork.locale' 属性来切换.

和Struts比较

Struts用户应该会熟悉使用application.properties资源包,在这个文件中你可以放置所有应用程序中要被翻译的信息.WebWork尽管把资源包拆成基于每个action或者model类的方式,但是你可能会在这些文件里不断重复文字信息.对此的一个快速修正方式就是创建一个叫ActionSupport.properties 的文件放在com/opensymphony/xwork 包中,并把它设置在classpath中.如果你的action都是ActionSupport的子类这会工作的很好.

Inversion of Control

This page last changed on Apr 25, 2006 by scud.

什么是控制反转 (IoC)?

控制反转是用来处理对象间的依赖关系. 为了了解控制反转(现在也称为依赖注入Dependency Injection), 请阅读 Martin Fowler关于IoC的文档.

推荐的IoC容器是Spring

伴随着其他特性,Spring同时也是IoC框架的一种. 在WebWork 2.2中, 它是被推荐使用的IoC容器. 一篇详细的描述如何集成Spring到WebWork中的文章请阅读 这里.
除了Spring以外, 还有许多其他容器可以使用, 比如Pico 和 (不赞成使用XWork的IoC容器)集成在XWork 中的IoC容器.

WebWork/XWork 集成的IoC容器

> ⚠ 在WebWork 2.2中, WebWork/XWork IoC 容器是不赞成使用的(虽然没有移除), WebWork 开发组推荐使用Spring作为你的IoC 容器

在WebWork Ioc中,拥有被管理的依赖(对象)的对象叫做"组件(component)".

- IoC综述
- Xwork组件架构 译注:老文档,不再翻译
- Webwork如何使用组件 译注:老文档,不再翻译
- 在Webwork和XWork中配置组件 译注:老文档,不再翻译

Components

This page last changed on Mar 30, 2006 by scud.

These documents are out of date. As of WebWork 2.2, the WebWork IoC container has been deprecated (though not removed) and the WebWork team recommends you use Spring for all your IoC needs

Overview

WebWork builds on XWork's component implementation by providing lifecycle management of component objects and then making these components available to your action classes (or any other user code for that matter) as required.

Two types of classes in WebWork can use an enabler interface for inversion of control: Actions and Components. In order for an Action class to have its components set, the ComponentInterceptor must be made available for the Action to set those resources. In turn, if those components require other components to be initialized and set for their own use, those initializations take place at the time the ComponentInterceptor intercepts the action as well.

Scopes and Lifecycle

Components can be configured to exist across three different scopes in WebWork:

1. for the duration of a single request,
2. across a user session, or
3. for the entire lifetime of the web application.

WW:WebWork lazy loads components, meaning that components, no matter what scope, are initialized at the time they are used and disposed of at the end of the given lifecycle of that scope. Thus, an application scoped component, for example, will be initialized the first time a user makes a request to an action that implements the enabler interface of that component and will be disposed of at the time the application closes.

While components are allowed to have dependencies on other components they must not depend on another component that is of a narrower scope. So, for example, a session component cannot depend on a component that is only of request scope.

All components must be registered in the components.xml file, which is discussed in the Configuration section.

Obtaining a ComponentManager

During any request there are three component managers in existence, one for each scope. They are stored as an attribute called "DefaultComponentManager" in their respective scope objects. So if for example you need to retrieve the ComponentManager object for the request scope, the following code will do the trick:

```
ComponentManager cm = (ComponentManager) request.getAttribute("DefaultComponentManager");
```

IoC Configuration

This page last changed on Mar 30, 2006 by scud.

These documents are out of date. As of WebWork 2.2, the WebWork IoC container has been deprecated (though not removed) and the WebWork team recommends you use Spring for all your IoC needs

Configuration - web.xml

To configure WebWork's component management, the following lines must be added in the appropriate places to web.xml:

```
<filter>
    <filter-name>container</filter-name>
    <filter-class>com.opensymphony.webwork.lifecycle.RequestLifecycleFilter</filter-class>
</filter>

<filter-mapping>
   <filter-name>container</filter-name>
   <url-pattern>*.action</url-pattern> <!-- modify appropriately -->
</filter-mapping>

<!-- Optionally you may instead apply the filter to EVERY URI instead of just *.action.  -->
<!-- You might want to do this for example:  -->
<!-- * your page flow goes to a jsp directly (as opposed to only *.action URIs) -->
<!-- * the jsp has an action in it to be run with the ww:action tag -->
<!-- * the action in the jsp implements any enabler interfaces to get components served to it
-->
<!-- The reason: (Per Patrick Lightbody) -->
<!-- "The components work by looking for a ComponentManager in the request -->
<!-- attributes. It gets placed there by a filter. If it is on *.action and a -->
<!-- request comes in through a JSP, the filter won't be applied and it will -->
<!-- never work." -->
<!-- The overhead in doing this is small, so don't worry overly much about the  -->
<!-- performance of this. -->
<!-- <filter-mapping> -->
<!--    <filter-name>container</filter-name> -->
<!--    <url-pattern>*</url-pattern>  -->
<!-- </filter-mapping> -->

<listener>
    <listener-class>com.opensymphony.webwork.lifecycle.SessionLifecycleListener</listener-class>
</listener>

<listener>
    <listener-class>com.opensymphony.webwork.lifecycle.ApplicationLifecycleListener</listener-class>
</listener>
```

These settings allow WebWork to manage components across the application, session and request scopes. Note that even if one or more of the scopes are not required by your application, all three scopes need to be specified in web.xml for WebWork's component management to function correctly.

Configuration - xwork.xml

The ComponentInterceptor class is used to apply the IoC pattern to XWork actions (ie, to supply components to actions). The ComponentInterceptor should be declared in the <interceptors> block of xwork.xml as follows:

```
<interceptor name="component"
        class="com.opensymphony.xwork.interceptor.component.ComponentInterceptor"/>
```

You should ensure that any actions that are to be supplied with components have this interceptor applied. (See OS:XWork Interceptors for information on how to apply interceptors to actions.)
If you want to apply IoC to objects other than actions or other components, you will need to use the ComponentManager object directly.

Note too, that the ComponentInterceptor is applied as part of the webwork defaultStack. Thus, if you are applying the defaultStack to the action, you would include the ComponentInterceptor.

Configuration - components.xml

The components.xml file is used to specify the components that are to be available. The components specified here are loaded into XWork's ComponentManager and are then made available to any actions that are an instance of the specified enabler. The components.xml file must be placed in the root of the classpath (ie, in the WEB-INF/classes directory).
Here is an example components.xml file that configures a Counter component. The Counter object will live in session scope, and will be passed to any objects that are enabled due to their implementing the CounterAware interface:

```
<components>
    <component>
        <scope>session</scope>
        <class>com.opensymphony.webwork.example.counter.Counter</class>
        <enabler>com.opensymphony.webwork.example.counter.CounterAware</enabler>
    </component>
</components>
```

Each component must have the following three attributes:

- scope: Valid values are application, session and request. This determines the component's lifetime. Application scope components will be created when the webapp starts up, and they will survive for the whole lifetime of the webapp. Session scoped components exist for the duration of a user session, while components in request scope only last for the duration of a single client request.
- class: This specifies the component's class. An instance of this object will live for the duration of the specified scope, and will be made available to any actions (or other code) as required. Note that components are lazy-loaded, so if nothing makes use of the component during its lifetime, the component will never actually be instantiated. At the moment components must have a zero argument constructor.
- enabler: Any actions that are an instanceof the enabler class or interface will be passed an instance of the component.

Note that while components are allowed to have dependencies on other components they must not depend on another component that is of a narrower scope. So for example, a session component cannot depend on a component that is only of request scope.

IoC Overview

This page last changed on Apr 04, 2006 by dogg.

⚠ 在WebWork 2.2中, WebWork/XWork IoC 容器是不赞成使用的(虽然没有移除), WebWork 开发组推荐使用Spring作为你的IoC 容器

综述

在许多应用中, 都有一些对象必需使用组件对象. 简单来说, IoC模式允许父对象(在Webwork和XWork中是ComponentManager实例)向需要的活动对象提供资源对象(通常是一个活动, 但也可以是任何实现适当的enabler接口的对象), 而不是需要对象自己获取资源.
有两种方式实现IoC: 初始化或使用enabler接口. 使用初始化方式, 给定活动对象使用资源对象作为构造函数的参数完成初始化. 使用enabler接口, 活动将具有接口的方法"setComponent(ComponentObject r);", 该方法允许在活动对象初始化后将资源对象传递到该对象. 根据对象实现的接口传递对应的组件对象. XWork使用enablers传递组件.

为什么使用IoC?

那么IoC为什么有用呢? 这意味着我们可以采用自顶向下方式开发组件(通常是某类服务, 如JDBC连接, 在客户程序中不需要知道连接是Oracle的还是Sybase的, 也不需要知道连接是从连接池获取还是其他方式), 而不需要构建一个注册类让客户调用它以获取组件实例.

实现服务的传统方式可能采用下面类似的步骤:

1. 编写组件(如ExchangeRateService)
2. 编写客户类(如XWork活动)
3. 编写保存该组件的注册类(如Registry)
4. 编写在注册类中注册指定组件对象的代码 (如Registry.registerService(new MyExchangeRateService()))
5. 在客户类中使用注册类获取服务(如, ExchangeRateService ers = Registry.getExchangeRateService())
6. 在客户类中调用组件对象(如, String baseCurrencyCode = ers.getBaseCurrency())

使用IoC, 该过程简化为:

1. 编写组件类(如ExchangeRateService)
2. 在XWork中注册组件类(如, componentManager.addEnabler(MyExchangeRateService, ExchangeRateAware))
3. 编写客户类, 并确保实现了enabler接口(如一个实现了ExchangeRateAware接口的XWork活动)
4. 在客户类中直接访问组件实例(如, String baseCurencyCode = ers.getBaseCurrency())

IoC的更多优点如下:

1. 可测试能力 - 更容易进行测试: 使用enabler方法向对象传递mock对象, 而不需要创建对象用于获取组件的容器.
2. 组件自描述. 当需要初始化一个组件时, 可以很容易得得知该组件需要那些依赖对象(组件)而不需要查看源代码, 文档或示例.
3. 使用反射(reflection)很容易发现依赖关系. 从图表生成到运行时优化都会带来好处(例如, 可以提前判断满足请求所需的组件并可以异步进行组件准备).
4. 避免使用超大工厂(super-uber-mega-factory)设计模式(应用的全部组件都保存在一个类中, 这直接导致与某些类的紧密绑定而难于'just use that one class').
5. 符合Demeter定律(Law of Demeter). 有人认为这很愚蠢, 但在实践中我发现这样工作的更好. 每个类仅和真正使用它的类耦合(and it should never use too much). 这种方法鼓励每个类承担更少的责任, 从而产生更清晰的设计.
6. 允许与环境(context)隔离并可以绕过. ThreadLocal在web应用中可能很合适hibernate样例代码中使用TnreafLocal管理session对象, 本文可能是针对这一情况, 但并不一定适合高并发的同步应用(如消息驱动的应用).

Xwork's Component Architecture

This page last changed on Mar 30, 2006 by scud.

These documents are out of date. As of WebWork 2.2, the WebWork IoC container has been deprecated (though not removed) and the WebWork team recommends you use Spring for all your IoC needs

Writing Component Classes

In XWork the actual component class can be virtually anything you like. The only constraints on it are that it must be a concrete class with a default constructor so that XWork can create instances of it as required. Optionally, a component may implement the Initializable and/or Disposable interfaces so it will receive lifecycle events just after it is created or before it is destroyed. Simply:

```
public class MyComponent implements Intializable, Disposable {
    public void init () {
         //do initialization here
    }

    public void dispose() {
         //do any clean up necessary before garbage collection of this component
    }
}
```

Component Dependencies

One feature that is not immediately obvious is that it is possible for components to depend on other components. For example if the ExchangeRateService described above depended on a Configuration component, XWork will pass the Configuration component through to the ExchangeRateService instance after ExchangeRateService is instantiated. Note that XWork automatically takes care of initializing the components in the correct order, so if A is an action or component that depends on B and C, and B depends on C and if A, B, and C have not been previously instantiated, the ComponentManager will in the following order:

1. Instantiate C and call it's init() method if it implements Initializable.
2. Instantiate B, then using the enabler method, set C to be used by B
3. Call B's init() method, if it implements Intitializable.
4. Set B using B's enabler method to be used by A.

And so on and so forth. Of course, if there are instances of B or C that would be reused in this case, those instances would be passed using the enabler method rather than a new instance.

Writing Enablers

An enabler should consist of just a single method that accepts a single parameter. The parameter class should either be the component class that is to be enabled, or one of the component's superclasses. XWork does not care what the name of the enabler's method is.

Here is an example of what the ExchangeRateAware enabler might look like:

```
public interface ExchangeRateAware {
    public void setExchangeRateService(ExchangeRateService exchangeRateService);
}
```

Note that typically an enabler would be an interface, however there is nothing to prevent you from using a class instead if you so choose.

Writing "Enabler-aware" Actions

All an action needs to do is implement the relevant enabler interface. XWork will then call the action's enabler method just prior to the action's execution. As a simple example:

```
public class MyAction extends ActionSupport implements ExchangeRateAware {
    ExchangeRateService ers;

    public void setExchangeRateService(ExchangeRateService exchangeRateService) {
        ers = exchangeRateService;
    }

    public String execute() throws Exception {
        System.out.println("The base currency is " + ers.getBaseCurrency());
    }
}
```

If you have an object that is not an action or another component, you must explictly tell XWork to supply any enabled components to your object by calling componentManager.initializeObject(enabledObject);

Using an external reference resolver

You can also use an external reference resolver in XWork, i.e., references that will be resolved not by XWork itself. One such example is using an external resolver to integrate XWork with the Spring Framework

You just need to write an external reference resolver and then tell XWork to use it in the package declaration:

```
<package
    name="default"
    externalReferenceResolver="com.atlassian.xwork.ext.SpringServletContextReferenceResolver">
```

Now, to use external references you do something like this:

```
<external-ref name="foo">Foo</external-ref>
```

Where the name attribute is the setter method name and Foo is the reference to lookup.

For more details and sample code about this integration, take a look at the javadocs to the com.opensymphony.xwork.config.ExternalReferenceResolver class (unfortunately unavailable online) and at XW-122

-Chris

J2SE 5 Support

This page last changed on May 18, 2006 by scud.

译注

由于Tiger支持还在开发中,所以很多类以及文档可能存在一些错误,所以请慎重使用,并参考源码.

使用WebWork中的J2SE 5 ("Tiger")

工作进行中

这一部分尚未完成! 以后会增加复杂的例子. 现在可以参考xwork-tiger项目目录下的单元测试作为例子.

要用WebWork中的J2SE5支持, 你需要把xwork-tiger.jar添加到classpath下. xwork-tiger.jar文件可以从ivy仓库的lib/tiger目录下获得.

拦截器Annotation

要使用这些Annotation, 需要在你的拦截器栈中加入AnnotationWorkflowInterceptor

Annotation	描述
After Annotation	标记一个需要在result之后执行的action方法.
Before Annotation	标记一个需要在主action方法之前执行的action方法.
BeforeResult Annotation	标记一个需要在result之前执行的action方法.

验证器Annotation

如果需要使用基于annotation的验证, 必须用Validation Annotation标注类或者接口

下面为XWork-tiger项目中的标准验证器Annotation

Annotation	描述
ConversionErrorFieldValidator Annotation	检查是否存在字段类型装换错误.
DateRangeFieldValidator Annotation	检查日期是否在指定范围内.
DoubleRangeFieldValidator Annotation	检查double类型字段是否在指定范围内.
EmailValidator Annotation	检查字段是否为e-mail地址.
ExpressionValidator Annotation	表达式检查.
FieldExpressionValidator Annotation	使用OGNL表达式检查.
IntRangeFieldValidator Annotation	检查数值字段是否在指定范围内.
RegexFieldValidator Annotation	正则表达式检查.
RequiredFieldValidator Annotation	检察字段是否为空.

RequiredStringValidator Annotation	检查String字段是否为空字符串.
StringLengthFieldValidator Annotation	检查String字段长度.
StringRegexValidator Annotation	检查String字段是否符合正则表达式.
UrlValidator Annotation	检查字段是否为合法URL.
Validation Annotation	标记验证为类型级别.
Validations Annotation	用验证annotation组.
VisitorFieldValidator Annotation	
CustomValidator Annotation	自定义验证器的annotation.

类型转换Annotation

如果xwork-tiger.jar文件已经在classpath下， 你可以直接通过泛型来为Map和Collection提供类型转换支持.

简言之， 使用 **泛型集合** 而不是在Type Conversion文档中指定集合和map的类型. 这就是说基本上不用 *ClassName-conversion.properties*文件了.

如果要使用基于annotation的类型转换， 你必须用Conversion Annotation标注类或者接口.

Annotation	描述
Conversion Annotation	在类型级别标注类型转换.
CreateIfNull Annotation	对于Collection和Map类型: 如果空则在Collection和Map里面创建新对象.
Element Annotation	对于泛型类型: 指定Collection和Map中的元素的类型.
Key Annotation	对于泛型类型: 指定Map中的key的类型.
KeyProperty Annotation	对于泛型类型: 指定键属性名字值.
TypeConversion Annotation	类型或者应用程序范围内的类型转换规则.

通过"ant apt"来创建ClassName-conversion.properties

这是一个使用apt ant target的例子:

```
<target name="apt">
        <mkdir dir="${build}/generated"/>

        <path id="classpath">
            <pathelement path="${basedir}/build/java"/>
            <pathelement path="${basedir}/build/test"/>
            <!-- xwork.jar and xwork-tiger.jar must be in one of the following lib dirs -->
            <fileset dir="${basedir}/lib/build" includes="*.jar"/>
            <fileset dir="${basedir}/lib/default" includes="*.jar"/>
            <fileset dir="${basedir}/lib/spring" includes="*.jar"/>
        </path>

        <property name="pclasspath" refid="classpath"/>

        <!-- Change the includes attribute value to match your annotated java files -->
        <fileset id="sources" dir="." includes="src/test/**/*.java" />

        <pathconvert pathsep=" " property="sourcefiles" refid="sources"/>
```

```
            <echo>

                CLASSPATH ${pclasspath}

                SOURCES: ${sourcefiles}

            </echo>

            <exec executable="apt" >
                <arg value="-s"/>
                <arg value="${build}/generated"/>
                <arg value="-classpath"/>
                <arg pathref="classpath"/>
                <arg value="-nocompile"/>
                <arg value="-factory"/>
                <arg value="com.opensymphony.xwork.apt.XWorkProcessorFactory"/>
                <arg line="${sourcefiles}"/>
            </exec>
        </target>
```

After Annotation

This page last changed on May 16, 2006 by scud.

After 标注

标注一个action的方法,在主action方法和result执行之后来调用.返回的值被忽略.

使用

After标准可以在方法级别上设置.

参数

没有参数

例子

```
public class SampleAction extends ActionSupport {

 @After
 public void isValid() throws ValidationException {
   // validate model object, throw exception if failed
 }

 public String execute() {
    // perform action
    return SUCCESS;
 }
}
```

This page last changed on May 08, 2006 by zjumty.

AnnotationWorkflowInterceptor 拦截器

调用action中的任何被标注的方法, 它支持一下特定的方法:

- @Before - 在action方法之前调用. 如果没有返回null, 则它的返回值作为action的结果码.
- @BeforeResult - 在action方法之后,result执行之前调用.
- @After - 在action方法和result执行之后被调用.

一个类中可以有多个方法有同样的标注,但是执行的顺序不一定. 尽管如此, 父类中的Before标注方法会在子类中Before标注方法之前调用,而After标注方法会在子类的After标注方法之后调用.

例子

```
public class BaseAnnotatedAction {
        protected String log = "";

        @Before
        public String baseBefore() {
                log = log + "baseBefore-";
                return null;
        }
 }

 public class AnnotatedAction extends BaseAnnotatedAction {
        @Before
        public String before() {
                log = log + "before";
                return null;
        }

        public String execute() {
                log = log + "-execute";
                return Action.SUCCESS;
        }

        @BeforeResult
        public void beforeResult() throws Exception {
                log = log +"-beforeResult";
        }

        @After
        public void after() {
                log = log + "-after";
        }
 }
```

配置xwork.xml把其中的PrepareInterceptor替换为AnnotationWorkflowInterceptor:

```
<interceptor-stack name="annotatedStack">
<interceptor-ref name="static-params"/>
<interceptor-ref name="params"/>
<interceptor-ref name="conversionError"/>
<interceptor-ref name="annotationInterceptor"/>
</interceptor-stack>
```

下面这一个AnnotatedAction,加上了AnnotationWorkflowInterceptor拦截器

```
<action name="AnnotatedAction" class="com.examples.AnnotatedAction">
   <interceptor-ref name="annotationInterceptor"/>
   <result name="success" type="freemarker">good_result.ftl</result>
</action>
```

随着这个拦截器的生效,AnnotatedAction中方法以后,log实例的值为baseBefore-before-execute-beforeResult-after

This page last changed on May 17, 2006 by scud.

Before 标注(Annotation)

标注一个action的方法,在主action方法之前执行.

使用

Before标注可以在方法级别设置.

参数

没有参数

例子

```
public class SampleAction extends ActionSupport {

 @Before
 public void isAuthorized() throws AuthenticationException {
   // authorize request, throw exception if failed
 }

 public String execute() {
    // perform secure action
    return SUCCESS;
 }
}
```

This page last changed on May 17, 2006 by scud.

BeforeResult 标注(Annotation)

标注一个action方法,在result之前需要被执行.返回值被忽略.

使用

BeforeResult 标注可以在方法级别上设置.

参数

没有参数

例子

```
public class SampleAction extends ActionSupport {

 @BeforeResult
 public void isValid() throws ValidationException {
   // validate model object, throw exception if failed
 }

 public String execute() {
    // perform action
    return SUCCESS;
 }
}
```

This page last changed on May 18, 2006 by scud.

类型转换Annotation

在Type级的类型转换Annotation标记

使用

Conversion标注必须应用于Type级

参数

参数	必须	默认值	描述
conversion	否		在Type级的类型转换Annotation标记

例子

```
@Conversion()
public class ConversionAction implements Action {
}
```

This page last changed on May 18, 2006 by scud.

ConversionErrorFieldValidator 标注

这个校验器检查是否有一个字段的任何转换错误,如果有则设置它们.浏览 类型转换错误处理 了解详细信息.

使用

ConversionErrorFieldValidator 必须用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.

例子

```
@ConversionErrorFieldValidator(message = "Default message", key = "i18n.key", shortCircuit =
true)
```

CreateIfNull Annotation

This page last changed on May 18, 2006 by scud.

CreateIfNull 标注

为类型转换设置 CreateIfNull

使用

CreateIfNull标注必须在字段级别上使用.

参数

参数	必填	缺省值	备注
value	no	false	CreateIfNull 属性值.

例子

```
@CreateIfNull( value = true )
private List<User> users;
```

CustomValidator 标注

这个标注可以用来定制校验器.使用 ValidationParameter 标注来添加额外的参数.

使用

这个标注必须设置在方法或者type级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型(ValidatorType)的枚举值(Enum value). 在这里可以是 FIELD 或者 SIMPLE.

例子

```
@CustomValidator(type ="customValidatorName", fieldName = "myField")
```

添加参数

使用 ValidationParameter annotation 来添加定制的参数值.

This page last changed on May 18, 2006 by scud.

ValidationParameter 标注

ValidationParameter 标注用来作为 CustomValidators 的参数.

使用

这个标准必须和CustomValidator 组合使用来提供一个参数.

参数

参数	必填	缺省	备注
name	yes		参数名称.
value	yes		参数值.

例子

```
@CustomValidator(
  type ="customValidatorName",
  fieldName = "myField",
  parameters = { @ValidationParameter( name = "paramName", value = "paramValue" ) }
)
```

This page last changed on May 18, 2006 by scud.

DateRangeFieldValidator 标注

这个校验器检查日期字段的值是否在指定范围内

使用

这个标注必须在方法级别上使用.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.
min	no		Date 属性. 日期允许的最小的值.
max	no		Date 属性. 允许的最大日期.

如果min以及max都没有设置,则任何事情都不会进行.

例子

```
@DateRangeFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
min = "2005/01/01", max = "2005/12/31")
```

DoubleRangeFieldValidator 标注

这个校验器检查double字段的值是否在一个指定的范围内.如果min和max都没有设置,则任何事情都不会发生.

使用

这个标注必须设置在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型(ValidatorType)的枚举值(Enum value). 在这里可以是 FIELD 或者 SIMPLE.
minInclusive	no		Double 属性. 最小允许的值(包含).
maxInclusive	no		Double 属性. 最大允许的值(包含).
minExclusive	no		Double 属性. 最小允许的值(不含).
maxExclusive	no		Double 属性. 最大允许的值(不含).

如果min和max都没有设置,则任何事情都不会发生.

min和max的值必须是当作一个字符串来插入,因此 "0"会被当作一个可能的值处理.

例子

```
@DoubleRangeFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
minInclusive = "0.123", maxInclusive = "99.987")
```

This page last changed on May 18, 2006 by scud.

Element 标注

设置类型转换的元素.

使用

Element标注必须使用在字段级别上.

参数

参数	必填	缺省值	备注
value	no	java.lang.Object.class	element 属性值.

例子

```
// The key property for User objects within the users collection is the <code>userName</code>
attribute.
@Element( value = com.acme.User )
private Map<Long, User> userMap;

@Element( value = com.acme.User )
public List<User> userList;
```

This page last changed on May 18, 2006 by scud.

EmailValidator 标注

这个校验器检查字段是否为一个合法的邮件地址,如果字段不是为空的话.

使用

这个标注使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.

例子

```
@EmailValidator(message = "Default message", key = "i18n.key", shortCircuit = true)
```

This page last changed on May 18, 2006 by scud.

ExpressionValidator 标注

这个非字段级别的校验器校验一个设置的正则表达式.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型(ValidatorType)的枚举值(Enum value). 在这里可以是 FIELD 或者 SIMPLE.
expression	yes		一个返回布尔值的OGNL表达式

例子

```
@ExpressionValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
expression = "an OGNL expression" )
```

This page last changed on May 18, 2006 by scud.

FieldExpressionValidator 标注

这个校验器使用一个OGNL表达式来进行校验.如果表达式在value stack求值后返回一个false则错误信息会被添加到此字段上.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型(ValidatorType)的枚举值(Enum value). 在这里可以是 FIELD 或者 SIMPLE.
expression	yes		一个返回布尔值的OGNL表达式

例子

```
@FieldExpressionValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
expression = "an OGNL expression")
```

This page last changed on May 18, 2006 by scud.

IntRangeFieldValidator 标注

这个校验器检查一个整数字段是否在指定范围内.如果min,max都没有指定,什么事情也不会发生.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.
min	no		Integer 属性. 允许的最小的值.
max	no		Integer 属性. 允许的最大的值.

如果min,max都没有指定,什么事情也不会发生.
min,max的值必须使用字符串来设置,因此"0"可以作为一个可能的值.

例子

```
@IntRangeFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true, min
= "0", max = "42")
```

This page last changed on May 18, 2006 by scud.

Key 标注

设置类型转换的Key

Usage

Key标注必须在字段级别上使用.

参数

参数	必填	缺省值	备注
value	no	java.lang.Object.class	key 属性值

例子

```
// The key property for User objects within the users collection is the <code>userName</code>
attribute.
@Key( value = java.lang.Long.class )
private Map<Long, User> userMap;
```

This page last changed on May 18, 2006 by scud.

KeyProperty 标注

为类型转换设置KeyProperty.

使用

KeyProperty标准必须设置在字段级别上.

这个标注应该和泛型一起使用,如果key元素的key属性需要指定的话.

参数

参数	必填	缺省值	备注
value	no	id	key 属性值.

例子

```
// The key property for User objects within the users collection is the <code>userName</code>
attribute.
@KeyProperty( value = "userName" )
protected List<User> users = null;
```

This page last changed on May 18, 2006 by scud.

RegexFieldValidator 标注

使用正则表达式校验一个字符串.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.

例子

```
@RegexFieldValidator( key = "regex.field", expression = "yourregexp")
```

This page last changed on May 18, 2006 by scud.

RequiredFieldValidator 标注

检查一个字段是否非空.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.

例子

```
@RequiredFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true)
```

RequiredStringValidator 标注

这个校验器检查字符串字段是否为空(例如,不是null而且长度>0).

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型(ValidatorType)的枚举值(Enum value). 在这里可以是 FIELD 或者 SIMPLE.
trim	no	true	Boolean 属性. 确定检查长度之前是否要trim字符串.

例子

```
@RequiredStringValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
trim = true)
```

This page last changed on May 18, 2006 by scud.

StringLengthFieldValidator 标注

这个校验器检查一个字符串字段是否在要求的长度内.它假设这个字段是一个字符串.如果minLength和maxLength都没有指定,任何事情都不会发生.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型(ValidatorType)的枚举值(Enum value). 在这里可以是 FIELD 或者 SIMPLE.
trim	no	true	Boolean 属性. 确定检查长度之前是否要trim字符串.
minLength	no		Integer 属性. 字符串最小允许长度.
maxLength	no		Integer 属性. 字符串最大允许长度.

如果minLength和maxLength都没有指定,任何事情都不会发生.

例子

```
@StringLengthFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
trim = true, minLength = "5",  maxLength = "12")
```

This page last changed on May 18, 2006 by scud.

StringRegexValidator 标注

StringRegexValidator 检查一个给定的字符串字段,如果不为空的话,是否匹配配置的正则表达式.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.
regex	yes	"."	String 属性. 要匹配的正则表达式.
caseSensitive	no	true	检查的表达式是否大小写敏感.

例子

```
@StringRegexValidator(message = "Default message", key = "i18n.key", shortCircuit = true, regex
= "a regular expression", caseSensitive = true)
```

TypeConversion Annotation

This page last changed on May 18, 2006 by scud.

TypeConversion 标注

这个标注用在类和应用范围上的转换规则.

类范围转换:
转换规则会在和相关的action类相同的包内的一个名称为 XXXAction-conversion.properties 的文件内获取.设置type为: type = ConversionType.CLASS

应用级别的转换:
这个转换规则会使用classpath根下的xwork-conversion.properties 文件内获取.设置type为: type = ConversionType.APPLICATION

使用

TypeConversion 标注可以设置在 property(属性)和方法级别上.

参数

参数	必填	缺省值	描述
key	no	被标注的属性/key名称	可选的属性名称,在TYPE级别标注时最长使用.
type	no	ConversionType.CLASS	ConversionType的枚举类型. 决定这个转换是应用级别的还是类级别的.
rule	no	ConversionRule.PROPERTY	ConversionRule的枚举类型. ConversionRule 可以是一个属性, 一个Collection 或者一个 Map.
converter	这个和value之一必填		用来进行转换的TypeConverter 的类名.
value	这个和 converter 之一必填		设置给ConversionRule.KEY_PROPERTY 的值.

例子

```
@Conversion()
public class ConversionAction implements Action {

  private String convertInt;

  private String convertDouble;
  private List users = null;

  private HashMap keyValues = null;
  @TypeConversion(type = ConversionType.APPLICATION, converter =
"com.opensymphony.xwork.util.XWorkBasicConverter")
  public void setConvertInt( String convertInt ) {
      this.convertInt = convertInt;
  }

  @TypeConversion(converter = "com.opensymphony.xwork.util.XWorkBasicConverter")
  public void setConvertDouble( String convertDouble ) {
      this.convertDouble = convertDouble;
  }

  @TypeConversion(rule = ConversionRule.COLLECTION, converter = "java.util.String")
  public void setUsers( List users ) {
      this.users = users;
  }

  @TypeConversion(rule = ConversionRule.MAP, converter = "java.math.BigInteger")
  public void setKeyValues( HashMap keyValues ) {
      this.keyValues = keyValues;
  }

  @TypeConversion(type = ConversionType.APPLICATION, property = "java.util.Date", converter =
"com.opensymphony.xwork.util.XWorkBasicConverter")
  public String execute() throws Exception {
      return SUCCESS;
  }
}
```

UrlValidator Annotation

This page last changed on May 18, 2006 by scud.

UrlValidator 标注

这个校验器检查字段是否是一个合法的URL.

使用

这个标注必须使用在方法级别上.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的 i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型 (ValidatorType)的枚举值 (Enum value). 在这里可以是 FIELD 或者 SIMPLE.

例子

```
@UrlValidator(message = "Default message", key = "i18n.key", shortCircuit = true)
```

Validation Annotation

This page last changed on May 18, 2006 by scud.

验证Annotation

如果你想使用基于标注的验证，你需要用验证Annotation标注类或者接口.

使用

验证标注必须应用于Type级

参数

参数	必需	默认值	备注
message	是		错误消息
key	否		语言文件中的键
fieldName	否		
shortCircuit	否	false	这个拦截器是否可以造成短路
type	也	ValidatorType.FIELD	ValidatorType的枚举类型.可以用FIELD或SIMPLE

例子

标注的接口

- 用@Validation()标记接口
- 在方法级上应用标注或者自定义标注

```
@Validation()
public interface AnnotationDataAware {

    void setBarObj(Bar b);

    Bar getBarObj();

    @RequiredFieldValidator(message = "You must enter a value for data.")
    @RequiredStringValidator(message = "You must enter a value for data.")
    void setData(String data);

    String getData();
}
```

标注的类

```
@Validation()
 public class SimpleAnnotationAction extends ActionSupport {
     @RequiredFieldValidator(type = ValidatorType.FIELD, message = "You must enter a value for
bar.")
     @IntRangeFieldValidator(type = ValidatorType.FIELD, min = "6", max = "10", message = "bar
must be between ${min} and ${max}, current value is ${bar}.")
     public void setBar(int bar) {
         this.bar = bar;
     }

     public int getBar() {
         return bar;
     }

     @Validations(
             requiredFields =
                     {@RequiredFieldValidator(type = ValidatorType.SIMPLE, fieldName =
"customfield", message = "You must enter a value for field.")},
             requiredStrings =
                     {@RequiredStringValidator(type = ValidatorType.SIMPLE, fieldName =
"stringisrequired", message = "You must enter a value for string.")},
             emails =
                     { @EmailValidator(type = ValidatorType.SIMPLE, fieldName = "emailaddress",
message = "You must enter a value for email.")},
             urls =
                     { @UrlValidator(type = ValidatorType.SIMPLE, fieldName = "hreflocation",
message = "You must enter a value for email.")},
             stringLengthFields =
                     {@StringLengthFieldValidator(type = ValidatorType.SIMPLE, trim = true,
minLength="10" , maxLength = "12", fieldName = "needstringlength", message = "You must enter a
stringlength.")},
             intRangeFields =
                     { @IntRangeFieldValidator(type = ValidatorType.SIMPLE, fieldName =
"intfield", min = "6", max = "10", message = "bar must be between ${min} and ${max}, current
value is ${bar}.")},
             dateRangeFields =
                     {@DateRangeFieldValidator(type = ValidatorType.SIMPLE, fieldName =
"datefield", min = "-1", max = "99", message = "bar must be between ${min} and ${max}, current
value is ${bar}.")},
             expressions = {
                 @ExpressionValidator(expression = "foo > 1", message = "Foo must be greater
than Bar 1. Foo = ${foo}, Bar = ${bar}."),
                 @ExpressionValidator(expression = "foo > 2", message = "Foo must be greater
than Bar 2. Foo = ${foo}, Bar = ${bar}."),
                 @ExpressionValidator(expression = "foo > 3", message = "Foo must be greater
than Bar 3. Foo = ${foo}, Bar = ${bar}."),
                 @ExpressionValidator(expression = "foo > 4", message = "Foo must be greater
than Bar 4. Foo = ${foo}, Bar = ${bar}."),
                 @ExpressionValidator(expression = "foo > 5", message = "Foo must be greater
than Bar 5. Foo = ${foo}, Bar = ${bar}.")
     }
     )
     public String execute() throws Exception {
         return SUCCESS;
     }
 }
```

This page last changed on May 18, 2006 by scud.

Validations 标注

如果你想使用几个相同类型的标注,这些标注必须嵌套在 @Validations() 标注内.

使用

在方法级别上使用.

参数

参数	必填	备注
requiredFields	no	在列表里添加 RequiredFieldValidators
customValidators	no	在列表里添加 CustomValidators
conversionErrorFields	no	在列表里添加 ConversionErrorFieldValidators
dateRangeFields	no	在列表里添加 DateRangeFieldValidators
emails	no	在列表里添加 EmailValidators
fieldExpressions	no	在列表里添加 FieldExpressionValidators
intRangeFields	no	在列表里添加 IntRangeFieldValidators
requiredFields	no	在列表里添加 RequiredFieldValidators
requiredStrings	no	在列表里添加 RequiredStringValidators
stringLengthFields	no	在列表里添加 StringLengthFieldValidators
urls	no	在列表里添加 UrlValidators
visitorFields	no	在列表里添加 VisitorFieldValidators
stringRegexs	no	在列表里添加 StringRegexValidator
regexFields	no	在列表里添加 RegexFieldValidator
expressions	no	在列表里添加 ExpressionValidator

例子

```
@Validations(
          requiredFields =
                  {@RequiredFieldValidator(type = ValidatorType.SIMPLE, fieldName =
"customfield", message = "You must enter a value for field.")},
          requiredStrings =
                  {@RequiredStringValidator(type = ValidatorType.SIMPLE, fieldName =
"stringisrequired", message = "You must enter a value for string.")},
          emails =
                  { @EmailValidator(type = ValidatorType.SIMPLE, fieldName = "emailaddress",
message = "You must enter a value for email.")},
          urls =
                  { @UrlValidator(type = ValidatorType.SIMPLE, fieldName = "hreflocation",
message = "You must enter a value for email.")},
          stringLengthFields =
                  {@StringLengthFieldValidator(type = ValidatorType.SIMPLE, trim = true,
minLength="10" , maxLength = "12", fieldName = "needstringlength", message = "You must enter a
stringlength.")},
          intRangeFields =
                  { @IntRangeFieldValidator(type = ValidatorType.SIMPLE, fieldName =
"intfield", min = "6", max = "10", message = "bar must be between ${min} and ${max}, current
value is ${bar}.")},
          dateRangeFields =
                  {@DateRangeFieldValidator(type = ValidatorType.SIMPLE, fieldName =
"datefield", min = "-1", max = "99", message = "bar must be between ${min} and ${max}, current
value is ${bar}.")},
          expressions = {
              @ExpressionValidator(expression = "foo > 1", message = "Foo must be greater than
Bar 1. Foo = ${foo}, Bar = ${bar}."),
              @ExpressionValidator(expression = "foo > 2", message = "Foo must be greater than
Bar 2. Foo = ${foo}, Bar = ${bar}."),
              @ExpressionValidator(expression = "foo > 3", message = "Foo must be greater than
Bar 3. Foo = ${foo}, Bar = ${bar}."),
              @ExpressionValidator(expression = "foo > 4", message = "Foo must be greater than
Bar 4. Foo = ${foo}, Bar = ${bar}."),
              @ExpressionValidator(expression = "foo > 5", message = "Foo must be greater than
Bar 5. Foo = ${foo}, Bar = ${bar}.")
  }
  )
  public String execute() throws Exception {
      return SUCCESS;
  }
```

This page last changed on May 18, 2006 by scud.

VisitorFieldValidator 标注

这个校验器允许你把校验器转向到使用对象自己的校验文件来校验你的action的对象属性.这允许你使用ModelDriver开发模式,在一个地方管理你的model的校验,也就是你的model类所在的地方.(译注:校验规则和类保存在相同目录下)

VisitorFieldValidator可以处理简单对象属性,对象集合或者数组. VisitorFieldValidator 的错误信息会通过对象的校验器添加到校验器信息的前面(The error message for the VisitorFieldValidator will be appended in front of validator messages added by the validations for the Object message.)

使用

这个标注必须在方法级别上使用.

参数

参数	必填	缺省值	备注
message	yes		字段错误信息
key	no		语言相关的属性文件里面的i18n key.
fieldName	no		
shortCircuit	no	false	校验器是否短路.
type	yes	ValidatorType.FIELD	校验器类型(ValidatorType)的枚举值(Enum value). 在这里可以是 FIELD 或者 SIMPLE.
context	no	action 别名	确定用来校验对象属性的上下文. 如果没有定义,校验的Action的上下文会传递给对象属性的校验.在Action校验这种情况下,上下文就是Action的别名.
appendPrefix	no	true	确定这个字段校验器的字段名是否应该加在要访问(visit)的字段的字段名前面,以便一个错误发生时来确定整个字段的名字.例如,假设被校验的bean有一个"name" 属性.如果appendPrefix设置为true,那么字段错误会保存在字段"bean.name"下.如果appendPrefix设置为false,那么字段错误会被保存在"name"字段下.

如果你使用 VisitorFieldValidator 来校验一个ModelDriven Action的model,你应该设置appendPrefix为false,除非你使用 "model.name"来引用你的model的属性.

例子

```
@VisitorFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
context = "action alias", appendPrefix = true)
```

JasperReports

This page last changed on Mar 28, 2006 by scud.

介绍

JasperReports(http://jasperreports.sourceforge.net)是领先的java开源报表库. 它将.jrxml(XML源文件)编译为.jasper(=编译后版本)文件, 它可以被转换为多种格式输出(PDF, CSV, XLS and HTML).
下面的例子, 我们使用Webwork动态创建人员列表的PDF文件. 我们的WW action用来创建对象的List, 而JasperReport Result用这个list来填充模版, 返回PDF文件.
我们假设你已经掌握WW web应用程序基本知识.

注意: 虽然例子非常简单, 我还是建议你详细的阅读WW和JR的相关文档.

使用的版本

Webwork 2.2 beta 3 (但是在以前的版本下也应该可以工作)
JasperReports 1.1.0
JDK 1.4.2

好的, 我们开始.

代码

我们先定义一个简单的POJO: Person.java

```
package com.mevipro.test;

public class Person {

        private Long id;

        private String name;

        private String lastName;

        public Person() {
                super();
        }

        public Person(String name, String lastName) {
                super();
                this.name = name;
                this.lastName = lastName;
        }



        public Person(Long id, String name, String lastName) {
                super();
                this.id = id;
                this.name = name;
                this.lastName = lastName;
        }

        /**
         * @return Returns the id.
         */
        public Long getId() {
                return id;
        }

        /**
         * @param id The id to set.
         */
        public void setId(Long id) {
                this.id = id;
        }

        /**
         * @return Returns the lastName.
         */
        public String getLastName() {
                return lastName;
        }

        /**
         * @param lastName The lastName to set.
         */
        public void setLastName(String lastName) {
                this.lastName = lastName;
        }

        /**
         * @return Returns the name.
         */
        public String getName() {
                return name;
        }

        /**
         * @param name The name to set.
         */
        public void setName(String name) {
                this.name = name;
        }


}
```

没什么特殊之处. 只有简单的属性, 构造方法, 还有getters和setters.

JasperReports库

在我们继续之前, 我们需要将JR库添加到classpath. 你可以从这里下载JR项目.
将jasperreports-X-project.zip存储到硬盘, 将文件解压缩.
我们需要如下文件:

- dist/jasperreports-X.jar
- lib/commons-*.jar (all the commons - except maybe for commons-logging)
- lib/itext-X.jar
- lib/jdt-compiler.jar

将这些jar拷贝到你的WW_WEBAPP/WEB-INF/lib目录, 然后将它们添加到你的classpath.

让我们看看Action

```
package com.mevipro.test.action;
```

```
import java.util.ArrayList;
import java.util.List;

import net.sf.jasperreports.engine.JasperCompileManager;

import com.mevipro.test.Person;
import com.opensymphony.xwork.ActionSupport;

public class JasperAction extends ActionSupport {

        //basic List - it will serve as our dataSource later on
 private List myList;

        /*
         * (non-Javadoc)
         *
         * @see com.opensymphony.xwork.ActionSupport#execute()
         */
        public String execute() throws Exception {

                // create some imaginary persons
         Person p1 = new Person(new Long(1), "Patrick", "Lightbuddie");
                Person p2 = new Person(new Long(2), "Jason", "Carrora");
                Person p3 = new Person(new Long(3), "Alexandru", "Papesco");
                Person p4 = new Person(new Long(4), "Jay", "Boss");

                /*
                 * store everything in a list - normally, this should be coming from a
                 * database but for the sake of simplicity, I left that out
                 */
                myList = new ArrayList();
                myList.add(p1);
                myList.add(p2);
                myList.add(p3);
                myList.add(p4);

                /*
                 * Here we compile our xml jasper template to a jasper file.
                 * Note: this isn't exactly considered 'good practice'.
                 * You should either use precompiled jasper files (.jasper) or provide some
kind of check
                 * to make sure you're not compiling the file on every request.
                 * If you don't have to compile the report, you just setup your data source
(eg. a List)
                 */
                try {
                        JasperCompileManager.compileReportToFile(
                                        "WW_WEBAPP/jasper/our_jasper_template.jrxml",
                                        "WW_WEBAPP/jasper/our_compiled_template.jasper");
                } catch (Exception e) {
                        e.printStackTrace();
                        return ERROR;
                }
                //if all goes well ..
         return SUCCESS;
        }

        /**
         * @return Returns the myList.
         */
        public List getMyList() {
                return myList;
        }

}
```

和刚才的代码一样--无需解释就很清楚了. 我们的JasperAction创建了一些人员的list. JasperCompileManager会将jrxml模版编译为.jasper文件.

不要在实际工作状态(production code)这样使用. 你当然应该提供编译后的模版文件, 或者做好变更检测, 来避免在每次请求时都重新编译模版. 但是在我们演示或者开发的过程中这样的方式很方便.

我们的Jasper模版

JR使用一种特殊的XML页面定义模版, 它会被编译为.jasper文件. 这些模版将会被用来设计结果报表. 它相当直接. 这是一个手写的版本 - 对于更加复杂的版本我强烈建议你看看各种各样的GUI设计器.

```
<?xml version="1.0"?>
    <!DOCTYPE jasperReport
  PUBLIC "-//JasperReports//DTD Report Design//EN"
  "http://jasperreports.sourceforge.net/dtds/jasperreport.dtd">
    <jasperReport name="jasper_test">
      <!-- our fields -->
      <field name="name" class="java.lang.String"/>
      <field name="lastName" class="java.lang.String"/>
      <title>
        <band height="50">
          <staticText>
            <reportElement x="0" y="0" width="180" height="15"/>
            <textElement/>
            <text>
              <![CDATA[Webwork JasperReports Sample]]>
            </text>
          </staticText>
        </band>
      </title>
      <pageHeader>
        <band></band>
      </pageHeader>
      <columnHeader>
        <band height="20">
          <staticText>
            <reportElement x="180" y="0" width="180" height="20"/>
            <textElement>
              <font isUnderline="true"/>
            </textElement>
            <text>
              <![CDATA[NAME]]>
            </text>
          </staticText>
          <staticText>
            <reportElement x="360" y="0" width="180" height="20"/>
            <textElement>
              <font isUnderline="true"/>
            </textElement>
            <text>
              <![CDATA[LASTNAME]]>
            </text>
          </staticText>
        </band>
      </columnHeader>
      <detail>
        <band height="20">
          <textField>
            <reportElement x="180" y="0" width="180" height="15"/>
            <textElement/>
            <textFieldExpression>
              <![CDATA[$F{name}]]>
            </textFieldExpression>
          </textField>
          <textField>
            <reportElement x="360" y="0" width="180" height="15"/>
            <textElement/>
            <textFieldExpression>
              <![CDATA[$F{lastName}]]>
            </textFieldExpression>
          </textField>
        </band>
      </detail>
      <columnFooter>
        <band></band>
      </columnFooter>
      <pageFooter>
        <band height="15">
          <staticText>
```

```
                <reportElement x="0" y="0" width="40" height="15"/>
                <textElement/>
                <text>
                  <![CDATA[Page:]]>
                </text>
              </staticText>
              <textField>
                <reportElement x="40" y="0" width="100" height="15"/>
                <textElement/>
                <textFieldExpression class="java.lang.Integer">
                  <![CDATA[$V{PAGE_NUMBER}]]>
                </textFieldExpression>
              </textField>
            </band>
          </pageFooter>
          <summary>
            <band></band>
          </summary>
        </jasperReport>
```

将文件存储到WW_WEBAPP/jasper/, 命名为'our_jasper_template.jrxml'.
最重要的:我们声明了name和lastName字段(不奇怪, 这两个属性来自我们的Person.class). 这意味着我们现在可以在我们的Jasper模版中使用这些字段.
我们定义了两个表头(NAME和LASTNAME), 然后将我们的字段们添加到一行的detail band(详细的解释请参照JR的教程).
'detail' band将会从人员的List中迭代. 这是JR的默认行为 - 所以如果你想显示人员的更多信息, 把它们添加到这个band中.
在detail band我们使用了

```
$F{name}
```

表达式. 这意味着JR会询问WW如何获取字段的值. 我们将会从WW值栈中寻找这些值(寻找人员, 调用getName()这个getter), 然后返回它. 后面的也一样

```
$F{lastName}
```

余下部分的大部分标记用来定义布局.

如果你遇到困难，在debug状态下通过logger(commons-logging, log4j, ..)来观察com.opensymphony.webwork.views.jasperreports可以方便寻找问题所在.

注册Action

好了, 可以将我们的action添加到xwork.xml了:

```
<action name="myJasperTest" class="com.mevipro.test.action.JasperAction">
        <result name="success" type="jasper">
                <param name="location">/jasper/our_compiled_template.jasper</param>
                <param name="dataSource">myList</param>
                <param name="format">PDF</param>
        </result>
</action>
```

我们进一步看一看. 我假设你已经熟悉了xwork的符号和schema, 如果你还不熟悉请查阅文档.

```
<action name="myJasperTest" class="com.mevipro.test.action.JasperAction">
```

我们将我们的JasperAction注册为'myJasperTest' - 这意味着我们可以在浏览器中通过myJasperTest.action发出请求来执行我们的Action.

```
<result name="success" type="jasper">
```

当我们的JasperAction执行正确， 我们使用注册为'jasper'的Result type. 如果你include了webwork-default, 它就已经被配置好了

```
<include file="webwork-default.xml"/>
```

这种result type根据我们的参数params配置, 配置如下:

```
<param name="location">/jasper/our_compiled_template.jasper</param>
```

这个参数定义了我们编译好的jasper文件的位置, 它将被WW根据我们的数据源dataSource填充:

```
<param name="dataSource">myList</param>
```

数据源的名称 - 就是你需要调用的getter的名字(上面的配置会调用你的JasperAction中的getMyList()方法). 它将被用来以数据填充模版.

```
<param name="format">PDF</param>
```

这一行制定了jasper被转换成的文件格式. 值可以是: PDF, CSV, XLS and HTML.

结论

你现在可以执行http://localhost:8080/YOUR_WEBAPP/myJasperTest.action - 然后你会看到一个不错的名字列表. WW提供了处理JasperReport文件的最优雅解决方案(也许); 指定.jasper文件的位置, 指定你希望使用的数据源dataSource, 然后它就可以工作了.

JSP

This page last changed on Mar 16, 2006 by scud.

Webwork中的JSP支持非常简单:缺省的webwork-default.xml将Dispatcher Result作为默认的result(参见Result Types). 这意味着任何JSP 1.2+ 容器可以直接支持Webwork.

快速上手

因为JSP支持通过默认的result Dispatcher Result触发,在配置xwork.xml时你不需要显式声明type属性:

```
<action name="test" class="com.acme.TestAction">
    <result name="success">test-success.jsp</result>
</action>
```

在 test-success.jsp 中:

```
<%@ taglib prefix="ww" uri="webwork" %>

<html>
<head>
    <title>Hello</title>
</head>
<body>

Hello, <ww:property value="name"/>

</body>
</html>
```

这里 name 是你的action中的一个属性.这样就可以了!

JUnit

This page last changed on Mar 30, 2006 by scud.

There's a number of approaches you can take to unit-test your WebWork actions.

The simplest is to instantiate your actions, call setters then execute(). This allows you to bypass all the complicated container setup.

Taken from Petsoar:

```
package org.petsoar.actions.inventory;

import com.mockobjects.constraint.IsEqual;
import com.mockobjects.dynamic.C;
import com.mockobjects.dynamic.Mock;
import com.opensymphony.xwork.Action;
import junit.framework.TestCase;
import org.petsoar.pets.Pet;
import org.petsoar.pets.PetStore;

public class TestViewPet extends TestCase {
    private Mock mockPetStore;
    private ViewPet action;

    protected void setUp() throws Exception {
        mockPetStore = new Mock(PetStore.class);
        PetStore petStore = (PetStore) mockPetStore.proxy();

        action = new ViewPet();
        action.setPetStore(petStore);
    }

    public void testViewPet() throws Exception {
        Pet existingPet = new Pet();
        existingPet.setName("harry");
        existingPet.setId(1);

        Pet expectedPet = new Pet();
        expectedPet.setName("harry");
        expectedPet.setId(1);

        mockPetStore.expectAndReturn("getPet", C.args(new IsEqual(new Long(1))), existingPet);
        action.setId(1);

        String result = action.execute();

        assertEquals(Action.SUCCESS, result);
        assertEquals(expectedPet, existingPet);
        mockPetStore.verify();
    }

    public void testViewPetNoId() throws Exception {
        mockPetStore.expectAndReturn("getPet", C.ANY_ARGS, null);

        String result = action.execute();

        assertEquals(Action.ERROR, result);
        assertEquals(1, action.getActionErrors().size());
        assertEquals("Invalid pet selected.", action.getActionErrors().iterator().next());
        assertNull(action.getPet());
        mockPetStore.verify();
    }

    public void testViewPetInvalidId() throws Exception {
        action.setId(-1);
        testViewPetNoId();
    }
}
```

Test interceptors and/or result types

Check out the test suites in XWork/WebWork. These are pretty comprehensive and provide a good starting point. For example, this is how the `ParametersInterceptor` is tested:

```
public void testDoesNotAllowMethodInvocations() {
    Map params = new HashMap();
    params.put("@java.lang.System@exit(1).dummy", "dumb value");

    HashMap extraContext = new HashMap();
    extraContext.put(ActionContext.PARAMETERS, params);

    try {
        ActionProxy proxy = ActionProxyFactory.getFactory().
               createActionProxy("", MockConfigurationProvider.MODEL_DRIVEN_PARAM_TEST,
extraContext);
        assertEquals(Action.SUCCESS, proxy.execute());

        ModelDrivenAction action = (ModelDrivenAction) proxy.getAction();
        TestBean model = (TestBean) action.getModel();

        String property = System.getProperty("webwork.security.test");
        assertNull(property);
    } catch (Exception e) {
        e.printStackTrace();
        fail();
    }
}
```

Note: these are not the ONLY ways so make your own judgement.

Misc

This page last changed on Mar 30, 2006 by scud.

This is a place for random pages that can't be classified anywhere else. Eventually they will be deleted if they can't be rolled in to a more appropriate location.

AJAX Validation - points for discussion

This page last changed on Mar 30, 2006 by scud.

WebWork 2 Ajax Validation

Validation servlet vs Interceptor

- Using a validation servlet requires the app developer to map any servlet filters for any action that is to be validated to the validation servlet as well.
- Using a validation interceptor executes the validation within the context of any mapped servlet filters, as well as within the context of the action.

Custom Ajax code or Third Party Library

- The DOJO Toolkit promises to be a very flexible and powerful toolkit, however they are only in the beginning stages. A lot of the code is coming from other dhtml toolkit projects that are very mature.
- We only require a fairly simple xmlhttp layer a the moment. Dojo have released their dojo.io.bind package http://dojotoolkit.org/intro_to_dojo_io.html which should be sufficient for us, however I think we can produce our own XmlHTTP code for now.
- I have implemented some custom xmlhttp / javascript code http://www.drivelater.com.au - do a search. It works in IE and FireFox.

Should webwork provide a static resource loader

- We need to provide a developer friendly way of exposing webwork static resources - primarily javascript files
  1. User could copy them into a folder
     - This either requires a separate zip download or packaging these files within the jar
     - This is a bad idea because it is error prone when the user upgrades
  2. Provide a resource loading servlet that serves the resources from the webwork jar
     requires a servlet definition and mapping in web.xml
     - could use a webwork.properties setting for the servlet prefix to allow user to 'mount' the static resources under a different prefix
       this makes jar upgrades easy and transparent
     - I have a prototype of this working in one of my apps - however it is restricted to mounting a single package on a single prefix. No support for multiple mappings.

```
i.e.  mount com.opensymphony.webwork.static at /webwork
request /webwork/validationAjax.js
loads  com.opensymphony.webwork.static.validationAjax.js
```

  3. Get the validationServlet to serve the javascript code. Use '/validationServlet/client.js' as the url
  4. Any other ideas ?

Supported Browsers

- We need to define what browsers we will support.

JavaScript API

- In webwork CVS /src/webapp/validationServlet.js contains a sample ValidationServlet client javascript class. It handles the communication with the validation servlet, and exposes callbacks for handling the errors. NOTE : I think we can re-work this a bit. I currently have lots of onWhatever callbacks. I think we should just have onErrors, and let the template designer do what they want with the Errors object.
    - Sample Usage

```
var validation = new ValidationServlet('/validationServlet/client.js');
validation.onErrors = function(inputObject, errors) {
        // clear old errors
 // display new errors
}
```

    - The errors param for the onErrors callback is a javascript object that has this structure

```
class Errors {
        String[] actionErrors;
        Map<String, String[]> fieldErrors; // fieldName is the key
}
```

- See this in action in webwork cvs head

```
/src/java/templates/xhtml/validation.vm
/src/webapp/validationServlet.js
/src/webapp/javascript-input.jsp
```

- Cloves' provided example API :

```
function addActionErrors(messages); // should messages be an array?!
function addFieldErrors(fieldName, messages); // should messages be an array?!
function clearActionErrors();
function clearFieldErrors(fieldName);
function clearErrors(formName);
```

New WebWork theme

- We need to develop a new slick looking template based on css that has full client side javascript support.
- We could mix some ideas from
    - http://www.themaninblue.com/experiment/InForm/margin.htm
    - http://www.baekdal.com/articles/Usability/usable-XMLHttpRequest/

CeWolf charts using Velocity templates

This page last changed on Mar 30, 2006 by scud.

Setup CeWolf

This currently only works with the most recent CVS version of WebWork but should be available in the upcoming 2.0 beta2

1. Go to http://cewolf.sourceforge.net and grab a stable release of CeWolf (at the time of writing, the unstable builds do not work with WebWork).
2. Edit your webwork.properties file and add "de.laures.cewolf.taglib.tags" to the property "webwork.velocity.tag.path"

Lastly add the CeWolf servlet to web.xml:

```
<servlet>
    <servlet-name>CewolfServlet</servlet-name>
    <servlet-class>de.laures.cewolf.CewolfRenderer</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>CewolfServlet</servlet-name>
    <url-pattern>/cewolf/*</url-pattern>
</servlet-mapping>
```

Create a DatasetProducer

This is the default DatasetProducer from the CeWolf tutorial.

```
import java.io.Serializable;
import java.util.Date;
import java.util.Map;

import org.jfree.data.DefaultCategoryDataset;

import de.laures.cewolf.DatasetProduceException;
import de.laures.cewolf.DatasetProducer;

public class PageViewCountData implements DatasetProducer, Serializable {

        // These values would normally not be hard coded but produced by
 // some kind of data source like a database or a file
 private final String[] categories =    {"mon", "tue", "wen", "thu", "fri", "sat", "sun"};
        private final String[] seriesNames =    {"cewolfset.jsp", "tutorial.jsp",
"testpage.jsp", "performancetest.jsp"};
        private final Integer[] [] values = new Integer[OS:seriesNames.length]
[OS:categories.length];

        public Object produceDataset(Map params) throws DatasetProduceException {
                DefaultCategoryDataset dataset = new DefaultCategoryDataset();
                for (int series = 0; series < seriesNames.length; series ++) {
                        int lastY = (int)(Math.random() * 1000 + 1000);
                        for (int i = 0; i < categories.length; i++) {
                                final int y = lastY + (int)(Math.random() * 200 - 100);
                                lastY = y;
                                dataset.addValue((double)y, seriesNames[OS:series],
categories[i]);
                        }
                }
                return dataset;
        }

        public boolean hasExpired(Map params, Date since) {
                return (System.currentTimeMillis() - since.getTime())  > 5000;
        }

        public String getProducerId() {
                return "PageViewCountData DatasetProducer";
        }
}
```

Create the Velocity template

With the new WebWork refactorings, nested JSP tags with arbitrary parameters can be used, so we convert the CeWolf tutorial JSP script to Velocity.

```
<jsp:useBean id="pageViews" class="de.laures.cewolf.example.PageViewCountData"/>
<cewolf:chart
    id="line"
    title="Page View Statistics"
    type="line"
    xaxislabel="Page"
    yaxislabel="Views">
    <cewolf:data>
        <cewolf:producer id="pageViews"/>
    </cewolf:data>
</cewolf:chart>

<cewolf:img chartid="line" renderer="cewolf" width="400" height="300"/>
```

In Velocity it looks like this:

```
#set( $pageViews = $stack.findValue("new com.PageViewCountData()") )
$req.session.setAttribute("pageViews", $pageViews )

#bodytag( SimpleChart "id=line" "title=Page View Statistics" "type=line" "xaxislabel=Page"
"yaxslabel=Views" )
  #bodytag( Data )
    #tag( Producer "id=pageViews" )
  #end
#end

#tag( ChartImg "chartid=line" "renderer=cewolf" "width=400" "height=300" )
```

As you may notice, CeWolf looks up it's DatasetProducer in the request attributes - it has no knowledge of the Velocity context. That's why we call $req.session.setAttribute(). The other attributes (such as the chartid) will be set by CeWolf, so we don't need to care about them.

Setup an action to disply the template

Now you should be able to fire up an action in the usual way with this template as the result and a nice chart should appear.

Developing WW Ajax Widgets

This page last changed on Mar 30, 2006 by scud.

This page is intended for WW developer not for WW users. If your a user of WW hopefully you dont need to know these details, but certainly more knowledge is better, so if your hungry eat up!

The WW ajax/dhtml components use the DOJO Toolkit for both javascript event handling and AJAX calls.

To understand the widget framework this page is a must read:
http://dojotoolkit.org/docs/fast_widget_authoring.html

To understand the dojo event framework which is used extensively by the widget framework read this:
http://dojotoolkit.org/docs/dojo_event_system.html

OK Great so your smart on DOJO Widgets and DOJO Events lets see how WW uses these.

WW Tag Code

Lets start with looking at what a user of WW would include in their page, a JSP in this case:

```
<ww:a
        id="link1"
        theme="ajax"
        href="/AjaxRemoteLink.action"
        showErrorTransportText="true"
        errorText="An Error ocurred">Update</ww:a>
```

This is just a standard WW component. All the attributes defined here must map to a component class thats defined in the taglib.tld. Were not going to go into how the WW componets work in this discussion, just be assured that nothing different happens here.

WW Component Template Code

WW via the component system will read in the ajax/a.ftl file to determine how to generate the following html file. Again this is standard WW Component processing so well breeze over this.

WW Generated Code

Now lets look at what this snippet of code would generate for us in html:

```
<a dojoType="BindAnchor" evalResult="true" id="link1" href="/ajax/AjaxRemoteLink.action"
errorHtml="An Error ocurred" showTransportError="true">Update</a>
```

Lets review this file. The dojoType type tells the dojo parser what to do with this tag. This is going to need to correspont to a call in our widge file like:
dojo.widget.tags.addParseTreeHandler("dojo:BindAnchor");
All the other attributes are basically passthrough from our component code to our widget code. No real magic happening w/ them.

Dojo Widget File

Next up lets look at some snippets of the widget file:

```
webwork.widgets.HTMLBindAnchor = function() {
//lots removed for clarity
 this.widgetType = "BindAnchor";
        this.templatePath = dojo.uri.dojoUri("webwork/widgets/BindAnchor.html");

        // the template anchor instance
 this.anchor = null;
//lots removed
         self.anchor.href = "javascript:{}";

                dojo.event.kwConnect({
                        srcObj: self.anchor,
                        srcFunc: "onclick",
                        adviceObj: self,
                        adviceFunc: "bind",
                        adviceType: 'before'
                });

//snippet removed
    }

}


// make it a tag
dojo.widget.tags.addParseTreeHandler("dojo:BindAnchor");
```

There are three aspects in this code snippet that are worth review. The declaration of the templatePath. This defines to DOJO where the prototype for its component lives. DOJO is going to read this component to determine what event binding needs to happen (TODO: a better description here). Were going to look at this template file in a moment, this file is where the real magic happens. The second aspect of this file is the kwConnect call... this is the ajax call to the remote source. I removed the snippet that happens afterwards but its not important here. And the final aspect worth understanding now is the last line. This tells dojo that this widget is registered as the name BindAchor... recall the dojoType="BindAnchor" in the above HTML snippet.

Dojo Template File

Now lets review the template file... BindAnchor.html... Understanding the template files were the big realization moment for me.

```
<a dojoAttachPoint="anchor"></a>
```

This template simply tells dojo one thing. the dojoAttachPoint. This attribute dojoAttachPoint is the javascript variable in the widget file to attach thid DOM object to. If you review the above widget code you will see a variable declaration called this.anchor = null; Dojo will plug this DOM object into that variable so its availabe in the widget JS. COOL.

OK... lets look at a more complex template file.

```
<input dojoAttachPoint="attachBtn" dojoAttachEvent="onClick: bind" type="submit" />
```

This is the code for the button template. Again its got the dojoAttachPoint attribute, but its also got dojoAttachEvent. This is based on the DOJO event system. the dojoAttachEvent property gives us a mapping between what the DOM Node's event name is (onClick) and what logical action to take as a result (bind).

This is how the the dojo widget code itercepts the button click code and calls the bind function.

Ok so now you can see how dojo intercepts calls to the real button click, and calls the bind funciton in your widget code. You could do most anything in this method. But in this case the bind function is defined in Bind.js which is the super JS inherited file to BindButton.js.

Hopefully this combined with the DOJO references is enough to get you up to speed.

DHTML and XmlHttpRequest References

This page last changed on Mar 30, 2006 by scud.

Javascript References

- Object Hierarchy and Inheritance in JavaScript - the best description of the OO nature of Javascript and inheritance in Javascript we've found

Opensource libraries / frameworks

- Dojo Toolkit An ambitious project started by the authors of several previous JS libraries. It aims to provide core utilities for XmlHttpRequest handling, event handling, accessibility / usability with back and forward button handling, etc. in addition to a set of rich UI widgets.
- DWR Direct Web Remoting - provides data binding and serialization between Java and Javascript
- Javascript Templates - allows you to define a text template much like Velocity, but in Javascript, then pass it the data to generate some text (HTML). Trimpath also has some other very cool stuff, but it's all GPL, whereas the JS Templates has been dual-licensed as Apache 2.0 as well.
- JSON-RPC Exposes Java objects for simple XmlHttpRequest remote invocation.
- Jsolait An OO framework for Javascript
- Tool-man examples - Great examples including drag-n-drop sorting and edit in place.

Examples / Sources

- The Ultimate JSP Tabs - Todd Ditchendorf's JSP taglib for tabbed panels and wizards
- Nifty Corners - very cool DHTML combination of CSS and Javascript to produce rounded corners without graphics. Also see More Nifty Corners.
- Using the XmlHttpRequest Object - one of the original articles on using XHR.
- Dynamic HTML and XML Another article on XHR from Apple
- Registering callbacks - examples of how to register callbacks for XmlHttpRequests
- Colorizing Java sources using Javascript and CSS - Interesting example of using Javascript and CSS to modify page content
- Fade Anything Technique Highlight changed items on a page with a colored background which fades away. See also the original Yellow Fade Technique.

Tools and Utilities

- Javascript Shell - a command line interface for Javascript and DOM
- Venkmann Javascript Debugger - Nice Javascript debugger for Mozilla / Firefox
- Selenium Javascript testing framework

Web Sites / Articles

- Ajaxian Frequently updated AJAX blog
- Ajax Matters AJAX blog and news aggregator

HttpSession in Action

This page last changed on May 19, 2006 by scud.

Use this if you want to be web agnostic (independent of the servlet API)

```
ActionContext.getContext().getSession()
```

The following gives you the same thing as above:

```
ServletActionContext.getRequest().getSession()
```

Note: Be sure not to use ActionContext.getContext() in the constructor of your action since the values may not be set up already (returning null for getSession()).

If you really need to get access to the HttpSession, use the ServletConfigInterceptor (see Interceptors).

Learn WW by example

This page last changed on Mar 30, 2006 by scud.

Table of contents: Learn WW by example

This guide is not yet ready for public consumption, and not yet linked to from the webwork wiki. Once enough pages have been written for it to become useful, I will link it into the wiki.

If you wish to contact me, please send email to rbondi at gmail d o t com.

/r:b:

Todo

1. how fit in both jsp and velocity in examples/toc?

Multiple Submit Buttons

This page last changed on Mar 30, 2006 by scud.

Introduction

Often, we have multiple submit buttons within a single form. The below is just a simple way of identifying which button was clicked, and which actions to take.

There are, of course, many ways of doing this, including the use of javascript to identify the actions, etc... You're welcome to pick and choose whichever method you find most useful. Webwork is flexible enough.

Form

```
<button type="submit" value="Submit" name="submit">
<button type="submit" value="Clear" name="clear">
```

Action with boolean properties

```
class MyAction extends ActionSupport {
   private boolean submit;
   private boolean clear;
   public void setSubmit(boolean submit) {
      this.submit = submit;
   }
   public void setClear(boolean clear) {
      this.clear = clear;
   }
   public String execute() {
      if (submit) {
         doSubmit();
         return "submitResult";
      }
      if (clear) {
         doClear();
         return "clearResult";
      }
      return super.execute();
   }
}
```

Explanation

The boolean properties 'submit' and 'clear' will be set to 'true' or 'false' according weather the submit or clear form element is present in the submitted form.

In this case, the properties are boolean, therefore the values set would be boolean.

There is another method, using String properties, described below...

Form

```
<button type="submit" value="Submit" name="buttonName">
<button type="submit" value="Clear" name="buttonName">
```

Action with String properties

```
class MyAction extends ActionSupport {
   private String buttonName;
   public void setButtonName(String buttonName) {
      this.buttonName = buttonName;
   }
   public String execute() {
      if ("Submit".equals(buttonName)) {
         doSubmit();
         return "submitResult";
      }
      if ("Clear".equals(buttonName)) {
         doClear();
         return "clearResult";
      }
      return super.execute();
   }
}
```

Explanation

In this case, the properties are String, therefore the values set are also String in nature.

I don't really like this method, as it ties in the Action to the Form. (What happens if you want different text to show up on the button ? You would have to change both the form as well as the corresponding action.)

Conclusion

There are other ways to achieve the same functionality. There are pros and cons to each methods. Feedback welcome.

Tabular inputs with HashMap

This page last changed on Mar 30, 2006 by scud.

Intro

I have a need to enter tabular data, like marks from a list of examination candidates.

This is how it's done :

the mark.vm file..

```
#foreach ( $candidate in $candidateList )
  #tag( TextField "label=" "name=marks['$candidate.id']" "value='$candidate.mark'" "size=3" )
#end
```

the SaveMarksAction

```
public class SaveMarksAction extends ActionSupport {
        private Map marks = new HashMap();

        public Map getMarks() {
                return marks;
        }


        public String execute() throws Exception {
                // get list of candidate IDs
         List candidateIds = marks.keySet();

                for (Iterator iter = candidateIds.iterator(); iter.hasNext();) {
                        String candidateId = (String) iter.next();
                        String mark = parseMap(marks.get(candidateId));

                        // process candidates and marks...
         }

        }

        // helper function to parse the map of entries....
 private static String parseMap(String[] map) {
                if (map == null) {
                        return null;
                }
                if (map.length != 1) {
                        return null;
                }
                return map[0];
        }


}
```

Explanation

The resulting vm file is rendered as

```
<input type="text" name="marks[OS:'candidateId1']" value="4" size="3"/>
<input type="text" name="marks[OS:'candidateId2']" value="5" size="3"/>
```

```
<input type="text" name="marks[OS:'candidateId3']" value="6" size="3"/>
<input type="text" name="marks[OS:'candidateId4']" value="7" size="3"/>
```

Webwork will populate the marks into the Map marks via

```
private Map marks = new HashMap();

public Map getMarks() {
        return marks;
}
```

whereby you can get the list of candidateIds via

```
List candidateIds = marks.keySet();
```

and the individual marks via

```
for (Iterator iter = candidateIds.iterator(); iter.hasNext();) {
        String candidateId = (String) iter.next();
        String mark = parseMap(marks.get(candidateId));
}
```

Possible enhancements

Couple tabular inputs with some sortable table component (javascript, client side)

- http://webfx.eae.net/dhtml/sortabletable/sortabletable.html

or

DisplayTag (server side)

- http://displaytag.sourceforge.net/
- http://www.displaytag.org/index.jsp

I believe there's some discussion on the mailing list about using Ognl to handle it automatically. I didn't follow it in detail, but from what I know, (do correct me if I'm wrong) the Ognl method is not available yet. The above works for now.

Conclusion

Feedback, comments and suggestions on better methods to perform the same function are welcome. If there's a simpler way, or a customised component to handle this tabular input automatically, I believe it'll be very useful.

OGNL

This page last changed on May 09, 2006 by tin.

概况

OGNL就是Object Graph Navigation Language的缩写(对象图形导航语言) - 可以从 http://www.ognl.org 获取OGNL的完整文档. 这篇文档中我们只演示与Webwork共存的一部分OGNL功能的例子. 如果想回顾基本的概念，请参考 OGNL基础.

Webwork使用了标准的上下文命名来进行OGNL表达式求值. OGNL处理的顶级对象是一个map (一般以context map(上下文map)引用). OGNL有一个根对象的概念 (在webwork中，它就是OGNLValueStack). 顺着根对象，其它对象被放置在context map中 (作为上下文引用)，包括你的 session/application/request/attr 这些map. 这些对象与根对象无关，它们只是存在于context map的一边 (保存在context map中). 所以，访问这些对象时需要使用 # 来告诉ongl不要在根对象中寻找，而是在其它的上下文中进行寻找.

```
                     |--request
                     |
                     |--application
                     |
       context map---|--OgnlValueStack(root)
                     |
                     |--session
                     |
                     |--attr
                     |
                     |--parameters
```

注意context map中还有其它的对象，我们在这个例子中只引用了一部分. 现在，你的action中的实例已经被保存在OGNLValueStack中，你可不用写 # 就可以引用这些bean属性了.

```
<ww: property value="myBean.myProperty"/>
```

对于 sessions,request,或者context map中的其它上下文:

```
ActionContext.getContext().getSession().put("mySessionPropKey", mySessionObject);
```

```
<ww:property value="#session.mySessionPropKey"/> or
<ww:property value="#session['mySessionPropKey']"/> or
<ww:property value="#attr.mySessionPropKey"/>
```

集合Collections (Maps, Lists, Sets)

在webwork中经常要处理集合类 (map, list, 和set)， 所以这里有一些使用select标签的例子:
list的语法: {e1,e2,e3}. 这样会创建一个包含String "name1", "name2" and "name3" 的List. 它还选择了 "name2" 作为默认值. 注意 Alt Syntax 的使用提供了字面意义 (literal) 上的 "name2".

```
<ww:select label="label" name="name" list="{'name1','name2','name3'}" value="%{'name2'}" />
```

map的语法: #{key1:value1,key2:value2}. 这样会创建一个map, 它将string "foo"影射到 string "foovalue", 将string "bar"影射到string "barvalue":

```
<ww:select label="label" name="name" list="#{'foo':'foovalue', 'bar':'barvalue'}" />
```

You may need to determine if an element exists in a collection. You can accomplish this with the operations `in` and `not in`
你可能需要确定一个元素是否存在于一个集合中. 你可以通过 `in` 和 `not in` 操作来实现.

```
<ww:if test="'foo' in {'foo','bar'}">
   muhahaha
</ww:if>
<ww:else>
   boo
</ww:else>

<ww:if test="'foo' not in {'foo','bar'}">
   muhahaha
</ww:if>
<ww:else>
   boo
</ww:else>
```

选择 (select) 一个集合的子集 (叫做投影), 你可以在collection中使用通配符.

- ? - 所有符合选择逻辑的元素
- ^ - 符合选择逻辑的第一个元素
- $ - 符合选择逻辑的最后一个一个元素

选择一个person (人) 对象的所有male relatives (男性亲属) 的子集:

```
person.relatives.{? #this.gender == 'male'}
```

Lambda表达式 (Lambda Expressions)

OGNL支持基本的lambda表达式语法, 允许你写一些简单的函数. 例如:

对于你们这些数学系的人，以为永远也不会再看到这个的人们.
Fibonacci(斐波那契序列): if n==0 return 0; elseif n==1 return 1; else return fib(n-2)+fib(n-1);
fib(0) = 0
fib(1) = 1
fib(11) = 89

有用的信息

lambda表达式就是括号中那部分. 这个 #this 变量代表这个表达式的参数, 它的初始值为11.

```
<ww:property value="#fib =:[#this==0 ? 0 : #this==1 ? 1 : #fib(#this-2)+#fib(#this-1)],
#fib(11)" />
```

OGNL Basics

This page last changed on May 09, 2006 by tin.

XWork专署语言功能

XWork在OGNL之上提供的最大改进就是支持ValueStack. OGNL操作假设这里只有一个 "根", XWork的ValueStack概念需要这里有很多 "根".

例如, 假设我们使用标准OGNL (不使用XWork), 这时如果有两个对象在 OgnlContext map 中: "foo" -> foo 而 "bar" -> bar, foo对象被配置为唯一的 **根(root)** 对象. 下面的代码演示了OGNL如何处理这三种情况:

```
#foo.blah // ## foo.getBlah()
#bar.blah // ## bar.getBlah()
blah      // ## foo.getBlah() ##foo####
```

着意味着OGNL允许上下文中保存多个对象, 但是除非你正在试图访问的这个对象是根对象, 否则必须预先跟上一个命名空间如 @bar. 现在让我们看看XWork, 它有一些不同之处...

有用的信息

在 XWork 中, 整个 ValueStack 是上下文的根对象. 不只根据你的表达式从栈 (stack) 中获取对象, 还从对象中获取属性 (例如: peek().blah), XWork有一个特殊的OGNL PropertyAccessor (属性访问器), 它可以自动搜寻栈内的所有实体 (从上到下), 直到找到一个具有与你所寻找的对象和属性匹配的.

例如, 假设栈中包括两个对象: Animal(动物)和Person(人). 两个多项都有 "name" 属性, Animal具有一个 "species" 属性, 而Person有一个 "salary" 属性. Animal是栈顶元素, Person在它后面. 下面的代码片断帮助你理解这个过程:

```
species    // ## animal.getSpecies()
salary     // ## person.getSalary()
name       // ## animal.getName() ##animal#####
```

最后一个例子, 这种关系让 animal 的 name 被返回. 一般这就是希望的效果, 但是有时你希望访问的是顺序靠后的对象的属性. 如果这样的话, XWork 还提供了按照索引 (index) 访问VlueStack的方法. 你只需这样做:

```
[0].name   // ## animal.getName()
[1].name   // ## person.getName()
```

通过表达式如 [0] ... [3] 等, WebWork将会截取栈的部分且返回一个 CompoundRoot(复合根) 对象. 获取那个特殊的部分栈 (stack cut), 使用0.top

ognl 表达式	描述
[0].top	将会返回这个部分栈 (stack cut)的栈顶元素, 也就是从整个栈的第0个开始 (这里与top相似)
[1].top	将会返回这个部分栈 (stack cut)的栈顶元素, 也就是从整个栈的第1个开始

访问静态属性

OGNL支持访问静态属性和静态方法. 如OGNL文档指出的, 你可以精确的调用静态内容, 通过如下方法:

```
@some.package.ClassName@FOO_PROPERTY
@some.package.ClassName@someMethod()
```

而且, XWork允许你不指定详细的类路径, 通过 "vs" 前缀调用你的action类中的静态属性或方法:

```
@vs@FOO_PROPERTY
@vs@someMethod()

@vs1@FOO_PROPERTY
@vs1@someMethod()

@vs2@BAR_PROPERTY
@vs2@someOtherMethod()
```

"vs" 代表 "value stack". 重要的是要注意如果你指定的类名仅仅是 "vs", 将使用栈顶对象的类. 如果你在 "vs" string后面指定了一个数字, 那么取而代之, 将使用栈中靠下面的对象的类.

与WebWork 1.x EL的区别

除了前面的例子和描述, 相对于WebWork 1.x 这里有一些主要的改变. 最大的改变是属性不再通过一个正斜杠 (/) , 而是使用 (.). 还有, 代替使用 ".." 来向上引用栈, 我们现在使用 "[n]" , 这里n是一个正数. 最后, 在WebWork 1.x 你可以通过 "@foo" 访问特别的命名对象 (准确的说是request scope的属性), 但是现在特殊的变量使用 "#foo"访问. 然而, 一定要注意 "#foo" **不再** 访问request属性. 因为 XWork 不只是为Web而设计的, 这里没有 "request属性" 这样的概念, 这样 "#foo" 仅仅是OgnlContext中除了根以外的另外一个对象.

老的表达式	新的表达式
foo/blah	foo.blah
foo/someMethod()	foo.someMethod()
../bar/blah	[1].bar.blah
@baz	不直接支持, 但是#baz是类似的
.	top 或 [0]

WebWork专署的命名对象

名称	值
#parameters['foo'] 或 #parameters.foo	request parameter ['foo'] (相当于 request.getParameter())
#request['foo'] 或 #request.foo	request attribute ['foo'] (相当于 request.getAttribute())
#session['foo'] 或 #session.foo	session attribute 'foo'
#application['foo'] 或 #application.foo	ServletContext attributes 'foo'
#attr['foo'] 或 #attr.foo	如果可以访问到则访问PageContext, 否则将分别访问 request/session/application 上下文!

Portlet Configuration

This page last changed on May 18, 2006 by tin.

XWork配置

你的应用程序的基本包(base package)应该扩展自 webwork-portlet-default 包, 例如:

```
<include file="webwork-default.xml" />

<package name="view" extends="webwork-portlet-default" namespace="/view">
```

Portlet初始化参数

下面是在 portlet.xml 中为portlet配置portlet模式(mode) -> xwork命名空间(namespace)影射的 init-param 元素. 简单的讲, 你可以把portlet模式理解为不同的子应用程序, 这样它可以为不同的portlet和portlet模式在 xowrk.xml 中配置不同的命名空间(namespace):

Key	描述	默认值
portletNamespace	这个portlet在xwork配置中的命名空间(namespace). 命名空间(namespace)将在查找action时被优先考虑, 从而允许在同一个portlet应用程序中建立(host)多个portlet. 如果设定了这个参数, 完整的命名空间(namespace)将会是 /portletNamespace/modeNamespace/actionName	默认命名空间.
viewNamespace	xwork配置中视图portlet模式(view portlet mode)的命名空间(namespace).	The default namespace.
editNamespace	xwork配置中编辑portlet模式(edit portlet mode)的命名空间(namespace). The namespace in the xwork config for the edit portlet mode.	默认命名空间.
helpNamespace	xwork配置中帮助portlet模式(help portlet mode)的命名空间(namespace).	默认命名空间.
defaultViewAction	在没有指定action名称时, 视图portlet模式(view portlet mode)使用的默认action名称.	默认值
defaultEditAction	在没有指定action名称时, 编辑portlet模式(edit portlet mode)使用的默认action名称.	默认值
defaultHelpAction	在没有指定action名称时, 帮助portlet模式(help portlet mode)使	默认值

	用的默认action名称.	

例子

```
<init-param>
    <!-- Portlet #### -->
    <name>portletNamespace</name>
    <value>/portletA</value>
</init-param>
<init-param>
    <!-- view portlet ####### -->
    <name>viewNamespace</name>
    <value>/view</value>
</init-param>
<init-param>
    <!-- #view portlet ########action## -->
    <name>defaultViewAction</name>
    <value>index</value>
</init-param>
```

这个 portlet.xml 的片断将会建立一个命名空间为 /portletA/ 的portlet. 这意味着所有对该portlet的请求将优先在此命名空间中寻找action. 进一步说, _视图(view) portlet 模式将会影射到 /view 命名空间, 这样一个对叫 myAction 的action的请求将会被转发到一个 /portletA/view/myAction 命名空间下的action上. 这也意味着如果没有请求一个action, 那么对这个请求将会调用默认的action index.

Portlet 阶段(phases)

Portlet规范描述一个portlet请求周期将持续为两个阶段, event(活动) 阶段 和 render(渲染) 阶段. 假设这个portlet中有 event(活动) 阶段, 那么它将肯定先于 render(渲染) 阶段执行. Event(活动) 阶段一般用来改变应用程序的状态. 在一个portlet中, 典型情况是在form提交的时候. Render(渲染) 阶段将会准备和分派到视图(view). 推荐你将一个在event(活动) 阶段执行的action的结果(result)指向到另外一个在 render(渲染) 阶段执行的action上, 后者用来负责派发到真正的视图.

Portlet 结果派发(result dispatching)

webwork-portlet-default 包定义了一个特殊的默认结果类型(result type), 它负责执行一个Action执行的结果逻辑(result logic). 一般, 这包括include一个jsp进行渲染, 或者为当前的活动 action准备一个渲染 action.

这个结果类型有三种主要执行模式.

- 如果这个Action在渲染阶段执行, 它会对配置在 location 属性下的资源执行 PortletRequestDispatcher.include(req, res) 方法.
- 如果这个Action在活动阶段执行, 并且结果是一个action影射(action mapping), 它会给ActionResponse设置一个渲染参数指定哪个Action将会在接下来的渲染阶段执行. 这符合良好的web应用程序设计, 着提促进了活动后重定向(redirect)的使用, 这样意味着一个在活动阶段执行的Action将会紧接着被重定向到一个在渲染阶段执行的Action.
- 如果这个Action在活动阶段被执行, 并且结果不是一个action影射(action mapping), 结果将会作为一种特殊的Action准备, 它被叫做 "renderDirect(直接渲染)" (在 webwork-portlet-default 包中指定), 它的唯一职责就是渲染特定的web资源 (一般是一个JSP).

在活动模式执行的action可以通过result配置中的query string给渲染模式执行的action传递渲染参数:

```
<result name="success">/displayCart.action?userId=${userId}</result>
```

这会将一个叫做 userId 的 渲染参数 值传递给将要派发到的action的 userId 属性.

Press Releases

This page last changed on Apr 25, 2006 by scud.

这里你会找到所有的WebWork的发布通讯稿.每个通讯稿都在顶部包含 关于,这样你无须关心 About了,只要你使用标准的通讯稿模板.

译注

- WebWork 2.x系列的发行公告以及其他发行公告可以在 Previous releases 页面里看到.
- 仅翻译了2.1.7和2.1.6的发布通讯稿,老的版本不在翻译

- 2.1.7 发布通讯稿
- 2.1.6 发布通讯稿
- 2.1.5 Press Release
- 2.1.4 Press Release
- 2.1.3 Press Release
- 2.1.2 Press Release
- 2.1.1 Press Release
- 2.1 Press Release

2.1 Press Release

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1 Released

The OpenSymphony group is proud to announce the release of WebWork 2.1. This release marks an important step in the continued development of the WebWork web application framework. The 2.1.x line will continue to be developed to add support to client side validation, stability, and performance. Future major versions of WebWork will be focusing on more client-oriented dynamic features (using DHTML and JavaScript), better tools integration (an IDEA plugin is in the works), and overall developer productivity.

- WebWork: http://www.opensymphony.com/webwork/
- Release notes and changes: http://www.opensymphony.com/webwork/wikidocs/Release%20Notes.html
- Documentation: http://www.opensymphony.com/webwork/wikidocs/Documentation.html

New Features

WebWork adds support for client side input validation, beefed up backwards compatibility, increased performance, and a new simplified UI layout theme ("simple"). The biggest addition, however, is in the form of documentation. The WebWork community came together and was able to create a great set of documentation and tutorials that will only get better over time. The OpenSymphony community has proved once again that it is one of the shining stars in the open source arena.

Bug Fixes

Many small bug fixes have been address, especially those relating to backwards compatibility with WebWork 1.x. In addition, the upgraded dependency on XWork 1.0.1 has increased performance greatly.

Special Thanks

The biggest addition to WebWork has, by far, been the documentation. Without the dedicated work from the community, WebWork would be nothing more than yet another excellent java library that no one but the authors use. Fortunately, the following people dedicated time and effort to making WebWork accessible by writing detailed documentation, examples, and tutorials:

- Vitor Souza
- Richard Hallier
- Bill Lynch
- Casey Moyaert
- Michael Campbell
- Gabriel Zimmerman
- Cuong Tran
- Michael Greer

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括 :Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

2.1.1 Press Release

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.1 Released

The OpenSymphony group is proud to announce the release of WebWork 2.1.1.

This release mostly fixes bugs that popped up in 2.1 as well as a couple new features. The full list of bug fixes and new features can be found in the release notes.

- WebWork: http://www.opensymphony.com/webwork
- Release notes and changes: http://www.opensymphony.com/webwork/wikidocs/Release%20Notes%20-%202.1.1.html
- Documentation: http://www.opensymphony.com/webwork/wikidocs/Documentation.html

New Features

- File upload support has been rebuilt to allow for multiple files with the same HTTP parameter name. Besides "cos" and "pell" support, "jakarta" support has been added, utilizing the Commons-FileUpload library. See the release notes for more detail.
- Validation now supports short-circuiting. This allows you to stop validation processing when a particular validation fails.

Bug Fixes

- You no longer must specify webwork.i18n.encoding in webwork.properties unless you wish to override the default value

Special Thanks

Special thanks to everyone who contributed bug reports and patches, and a very special thanks to Mark Woon for picking up the slack. Also special thanks to Bruce Ritchie of Jive Software for updating file upload support.

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括

:Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

2.1.2 Press Release

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.2 Released

The OpenSymphony group is proud to announce the release of WebWork 2.1.2.

- WebWork: http://www.opensymphony.com/webwork/
- Release notes and changes: http://www.opensymphony.com/webwork/wikidocs/Release%20Notes%20-%202.1.2.html
- Documentation: http://www.opensymphony.com/webwork/wikidocs/Documentation.html
- Download: https://webwork.dev.java.net/files/documents/693/7888/webwork-2.1.2.zip

Bug Fixes

- WebWork 2.1.2 comes packages with XWork 1.0.3, which fixes a simple but painful bug that many people experienced. It is recommended that all users upgrade to WebWork 2.1.2 and use the libraries that are shipped with it.

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括 :Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

2.1.3 Press Release

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.3 Released

The OpenSymphony group is proud to announce the release of WebWork 2.1.3.
This release fixes a critical bug found in WebWork 2.1.2. All users should download this release. No other changes have been made, so upgrading the webwork jar file is all that needs to happen.

- WebWork: http://www.opensymphony.com/webwork/
- Release notes and changes: http://www.opensymphony.com/webwork/wikidocs/Release%20Notes%20-%202.1.3.html
- Documentation: http://www.opensymphony.com/webwork/wikidocs/Documentation.html
- Download: https://webwork.dev.java.net/files/documents/693/7935/webwork-2.1.3.zip

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括 :Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

2.1.4 Press Release

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.4 Released

The OpenSymphony group is proud to announce the release of WebWork 2.1.4. This release is almost exactly like 2.1.3 except that it introduces a new JSP tag syntax. This syntax is optional and will disabled by default for the entire 2.1.x releases. Please Read the release notes for more information

- WebWork: http://www.opensymphony.com/webwork/
- Release notes and changes: http://www.opensymphony.com/webwork/wikidocs/Release%20Notes%20-%202.1.4.html
- Documentation: http://www.opensymphony.com/webwork/wikidocs/Documentation.html
- Download: https://webwork.dev.java.net/files/documents/693/8009/webwork-2.1.4.zip

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括 :Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

2.1.5 Press Release

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.5 Released

The OpenSymphony group is proud to announce the release of WebWork 2.1.5.
This release adds several key enhancements to the UI tag libraries, including more JavaScript event handlers, XHTML compatibility, and bug fixes.

- WebWork: http://www.opensymphony.com/webwork/
- Release notes and changes: http://www.opensymphony.com/webwork/wikidocs/Release%20Notes%20-%202.1.5.html
- Documentation: http://www.opensymphony.com/webwork/wikidocs/Documentation.html
- Download: https://webwork.dev.java.net/files/documents/693/8164/webwork-2.1.5.zip

Special Thanks

A very special thanks to Mathias Bogaert for helping out with all the UI tag library enhancements and bug fixes.

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括:Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

2.1.6 Press Release

This page last changed on Apr 25, 2006 by scud.

WebWork 2.1.6 发布了

OpenSymphony 组织自豪地宣布WebWork 2.1.6的发布.

- WebWork: http://www.opensymphony.com/webwork/
- 发行公告和改变: http://www.opensymphony.com/webwork/wikidocs/Release%20Notes%20-%202.1.6.html
- 文档: http://www.opensymphony.com/webwork/wikidocs/Documentation.html
- 下载: https://webwork.dev.java.net/files/documents/693/8838/webwork-2.1.6.zip

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括 :Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

2.1.7 Press Release

This page last changed on Apr 25, 2006 by scud.

WebWork 2.1.7 发布了

OpenSymphony 组织自豪地宣布WebWork 2.1.7的发布. 这个发布集中在小的UI框架的改进,包括一个新的强大的客户端校验(目前还在原型状态).

- WebWork: http://www.opensymphony.com/webwork
- 发行公告和改变: http://www.opensymphony.com/webwork/wikidocs/WebWork%202.1.7.html
- 文档: http://www.opensymphony.com/webwork/wikidocs/Documentation.html
- 下载: https://webwork.dev.java.net/files/documents/693/9723/webwork-2.1.7.zip

特别感谢

WebWork团队感谢下面的人们,因为他们对错误报告,用户支持,文档和补丁的帮助:

- Mathias Bogaert 修正了很多JIRA 中的问题
- Casey Moyaert 处理了大多数的文档并继续让它变得更好

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括 :Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

About

This page last changed on Apr 05, 2006 by scud.

关于 WebWork

WebWork是一个流行的开源Java web应用程序框架.最初由 Rickard Oberg (JBoss的原开发人员和XDoclet的创建者,还有很多建树) 开发,WebWork的目标是降低开发web应用程序的门槛,让更多的web开发的单调无味的任务自动化.汲取现存的其他web框架的最优特性,通过活跃的OpenSymphony社区的反馈,WebWork代表了web开发的最优解决方案.

WebWork构建在XWork之上,它是一个普通的command模式的框架.WebWork利用XWork的能力来提供下列特性:

- 高级的UI组件,允许你构建复杂,可复用的UI组件,从简单的文本输入框到高级的日期选择器.
- 一个健壮的控制反转容器,绑定到原生的Servlet生命周期上:request,session和application.
- 可插入的配置,允许你开发web "模块",可以轻松的结合在一起形成独立完整的应用程序.
- 完整的从HTTP到Java数据对象的数据映射,允许你把更多精力集中在应用程序开发上,更少的关系单调的数据转换.
- 一个完整的校验框架,包括服务器端和客户端.这让你选择最佳的方式在处理数据之前来保证用户的输入是正确的.
- 一种高级的表达式语言,基于OGNL,提供通常和构建基于web的用户界面相关的最通用的操作.
- 提供和很多流行的开源项目的集成,包括:Spring,Pico,OSWorkflow,FreeMarker,Velocity,JasperReports,JFreeChart,和其他非常多的.

Previous releases

This page last changed on Apr 25, 2006 by scud.

- 发行公告
  - 2.2.2发行公告
  - WebWork 2.2.1
  - WebWork 2.2

老版本的发行公告和升级指导

- 发行公告
  - WebWork 2.1.7
  - Release Notes \- 2.1.6
  - Release Notes \- 2.1.5
  - Release Notes \- 2.1.4
  - Release Notes \- 2.1.3
  - Release Notes \- 2.1.2
  - Release Notes \- 2.1.1
  - Release Notes \- 2.1
- 从前一个版本升级
  - Upgrading from 2.1.5
  - Upgrading from 2.1.4
  - Upgrading from 2.1.3
  - Upgrading from 2.1.2
  - Upgrading from 2.1.1
  - Upgrading from 2.1
  - Upgrading from 2.0
  - Upgrading from 1.4

Release Notes - 2.1

This page last changed on Mar 30, 2006 by scud.

Key Changes

- JavaScript client validation support - not totally complete, but basic validators work well. Look at the validators.xml file include in src/example to see how you can configure your validators to do client side validation on top of their normal duties
- The label attribute in UI tags are no longer required
- The themes and templates in UI tags behave like they did in 1.x
- A new theme, in addition to the existing "xhtml" one, called "simple" is included that doesn't have any of the labels, error reporting, or table rows that the "xhtml" template has. This is more in line with the tags included with Struts.
- New UI tags for CSS styles and classes added: cssStyle and cssClass
- Old action!command URL support works again. This means you can invoke a doCommand() method like in 1.x
- ww:param tag no longer requires the name attribute (for ordered params, like with ww:text). It also evaluates the the body as the value if no value is given.
- UI tags now have access to the FormTag parameter map using the "form" key. This means $parameters.form.name would return the form name, for example. The result is that complex JavaScript-based components can be built.

Migration Notes

Version	Description	Old Code	New Code
2.0	WebWorkUtil has been refactored into a number of classes, and the constructor has changed. If you were using it for Velocity support before, look at VelocityWebWorkUtil now		
2.0	The webwork.ui.templateDir configuration property has been broken into webwork.ui.templateDir and webwork.ui.theme	`webwork.ui.templateDi /webwork/mytheme`	`webwork.ui.templateDir = /webwork webwork.ui.theme = mytheme`
2.0	"namespace" attribute of the ww:action tag is now evaluated; those upgrading from 2.0 will need to place single quotes around the attribute value	<ww:action namespace="/foo" .../>	<ww:action namespace="'/foo'" .../>
2.0, but not 1.x	theme and template attributes in UI tags have changed are now evaluated; those upgrading from 2.0 will need to place single quotes around the attribute value	<ww:xxxx theme="/template/foo" template="bar.vm"/>	<ww:xxxx theme="'foo'" template="'bar.vm'"/>
1.x, 2.0	label UI tag evaluates the value attribute now	<ww:label name="'Foo'"/>	<ww:label value="'Foo'"/>

	instead of the name attribute		

Changelog

OpenSymphony JIRA (25 issues)		
T	Key	Summary
↗	WW-592	Upgrade commons-logging
●	WW-560	SessionMap holds on to requests when it doens&apos;t need to
+	WW-546	Make the config-browser show validators applied via the XML validation files
●	WW-544	Velocity result hardcodes contenttype and encoding
↗	WW-541	Webpage link for download
●	WW-537	Velocity tag outputs to the response, not the velocity writer
●	WW-530	Config Browser doesn&apos;t work after lates ActionConfig refactoring
●	WW-519	ActionTag should evaluate namespace attribute
↗	WW-518	Label attribute shouldn&apos;t be required
●	WW-517	Themes and templates should behave like 1.x
↗	WW-516	Simple theme that has no tables and xhtml extends from
●	WW-515	Class attribute is illegal
●	WW-514	Form tag double evaluates name attribute
●	WW-503	Fix tag libraries
●	WW-502	foo!default.action should work
+	WW-501	JavaSript-based client side validation
●	WW-500	ww:param tag fixes
+	WW-499	UI tags should have access to form
▣	WW-488	Check QuickStart Guide to make sure it works
●	WW-487	WebWorkConversionErrorInterceptorTest in wrong branch
●	WW-484	label tag problems
●	WW-478	URLTag tld entry does not correspond with actual property
↗	WW-476	WebWork needs a simple changelog for each release
●	WW-475	Multipart encoding still not fixed
●	WW-474	Ability to dynamically create array of Objects from a given request

Release Notes - 2.1.1

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.1 Release Notes

Key Changes

- Improved integration with Sitemesh
    - WebWork taglibs can be used in Sitemesh decorators to access Action properties
- Validator short-circuiting to allow validation to stop on first invalid data
- Improved class hierarchy resource bundle searching
- File upload support has been rebuilt to allow for multiple files with the same HTTP parameter name. Besides "cos" and "pell" support, "jakarta" support has been added, utilizing the Commons-FileUpload library. Only "jakarta" supports multiple files with the same HTTP parameter name. In future versions "jakarta" may become the default upload library, replacing "pell".

Migration Notes

Version	Description	Old Code	New Code
2.1	There is a new validator DTD: xwork-validator-1.0.2.dtd. You aren't required to use this, but you will need to if you wish to use the new short-circuiting validation	N/A	N/A
2.1	File upload support has been rebuilt, although we don't see any compatibility problems with 2.1. However, many of the methods in MultiPartRequest have become deprecated in favor of new ones. Please switch to these as soon as possible.	N/A	N/A

Changelog

WebWork 2.1.1

OpenSymphony JIRA (25 issues)

T	Key	Summary
●	WW-1078	Broken links in Shopping-Cart
●	WW-1077	setting theme in page, context, or session scope has not effect

☒	WW-1076	Move all Velocity templates to archive
☒	WW-1075	url validator docs
☒	WW-1074	AJAX docs screwed up
☒	WW-1073	control tags docs snippets are borken
●	WW-1072	select.ftl emptyOption evaluation
●	WW-1071	DoubleSelect&apos;s second select does not allow a predefined value to be selected
●	WW-1070	double select created option with key and value wrongly
↗	WW-1069	Ability to use freemarker map built-ins (?keys, ?values) as well as plain map methods (.keySet(), get(foo))
↗	WW-1065	TestCase for the Property component
☒	WW-1064	have a link in tags page to altsyntax page
✚	WW-1062	Add a OptionTransferSelect component
●	WW-1061	checkboxlist.ftl does not allow disabled parameter handling in WW2.2beta4
☒	WW-1059	Make the dojo configuration in ww:head hook into the i18n infrastructure
●	WW-1058	xhtml theme&apos;s form-validate.ftl onsubmit javascript is bad
●	WW-1057	DefaultActionMapper appending extra slash when namespace is "/"
↗	WW-1056	Change defaultStack order
↗	WW-1055	Warn when properties are null
☒	WW-1054	Include common build in project
☒	WW-1053	AJAX Validation Documentation
●	WW-1052	Always cleanup ActionContext
↗	WW-1051	Enable tag attribute description to be usable for both javadoc and tagdoclet
☒	WW-1050	Add DWR 1.1-beta2 jar to ivy-repository
●	WW-1049	ChainingInterceptor doesn&apos;t accept null of CompoundRoot element

Xwork 1.0.2

OpenSymphony JIRA (15 issues)

T	Key	Summary
↗	XW-210	Make default type conversion message a localized text that can be overidden
●	XW-205	missing xwork 1.0.2 dtd in jar and website and typo in ValidationInterceptor

↗	XW-204	TextProvider.getText() should look in child property files
↗	XW-203	Add "trim" parameter to string validators
●	XW-202	Integer and Float conversion dont work in CVS HEAD
●	XW-200	i18n broken when the name of the text to find starts with a property exposed by the action
↗	XW-195	Add interface XWorkStatics which contains XWork-related constants from WebWorkStatics
↗	XW-194	Patch to help LocalizedTextUtil deal with messages for indexed fields (collections)
↗	XW-193	InstantiatingNullHandler and Typeconversion fails
↗	XW-192	Create a version 1.0.2 of the XWork validation DTD with short circuit
↗	XW-191	Type conversion improvement.
↗	XW-190	Provide a xwork-default.xml.
↗	XW-189	Improve ActionValidationManager&apos;s short circuit behaviour
↗	XW-179	Optimise OgnlUtil.copy method
+	XW-172	XWorkBasicConverter doesn&apos;t care about the current locale

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.2 Release Notes

Key Changes

- This version ships with XWork 1.0.3 - we recommend you make sure you are running this version (or later) of XWork.
- Minor bug fixes for file upload support with Jakarta
- New StreamResult type which allows you to stream content directly back from an action
- UI tags may now be written in languages other than Velocity. JSP is supported, though you currently must write your own templates similar to the Velocity templates. Future versions of WebWork will include more languages supported as well as templates shippped out of the box.

Migration Notes

Migration should require nothing more than copying over the new libs. Specifically note that XWork 1.0.3 and WebWork 2.1.2 should be copied over.

Changelog

OpenSymphony JIRA (14 issues)

T	Key	Summary
↗	WW-642	Allow the &apos;name&apos; attribute of the TextTag to be evaluated at runtime
●	WW-639	"Could not open template ", possible a bug
↗	WW-634	File Upload Interceptor stack
●	WW-633	Jakarta File Upload fails with mixed content (normal and file)
➕	WW-630	upload newest webwork files to ibiblio
↗	WW-629	checkboxlist doesn&apos;t have a disabled attribute tag
●	WW-628	Bug with request parameter handling with WebLogic 8.1sp3
●	WW-624	If/Else tag do not render body
●	WW-622	The changelog for WW2.1.1 shows open issues not the closed ones!
●	WW-616	ww:label and ww:textarea problem with null values
●	WW-612	WebworkStatistics.SERVLET_DISPATCHER is spelled incorrectly
●	WW-611	Error in Freemarker docs
➕	WW-602	Stream Result Type
▣	WW-485	Add docs for WebWork2 tags

Release Notes - 2.1.3

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.3 Release Notes

Key Changes

WebWork version 2.1.3 resolves a critical problem preventing UI tags from working under certain circumstances. It is recommended that all users use 2.1.3 in place of 2.1.1 or 2.1.2

Changelog

OpenSymphony JIRA (2 issues)

T	Key	Summary
●	WW-659	VelocityTemplateEngine broken
➕	WW-321	ActionMessage as the companion of ActionError

Release Notes - 2.1.4

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.4 Release Notes

Key Changes

This release is the first release to include the re-vamped JSP tag support. Specifically, there is an option (off by default) that now lets you use an alternative syntax for JSP tags.

When **webwork.tag.altSyntax** is set to true in webwork.properties, all attributes in the JSP tags (both UI and non-UI) that evaluate to a String (as opposed to a Boolean, Integer, Collection, or anything else) shall not be evaluated like it normally is.

Rather, the string will be parsed for the pattern "%{...}" and only the text between those braces shall be evaluated. This should make using all the tags, but especially the UI tags, much easier.

The only exception to this rule of parsing String attributes is for the <ww:property/> tag. That is because the usage for <ww:property/> is so commonly used for pulling values from the stack, enforcing the "%{..}" syntax on it would be overly tedious.

Migration Notes

There is nothing to migrate for now. Because this new syntax is optional and turned off by default, you don't need to do anything to migrate as long as you don't plan to use this new syntax. If you DO plan to use this new syntax, you must modify all your tag attributes that previously had the pattern of "'...'" to just be "...". Basically, you must remove single quotes where they once were.

Also, any place where an attribute did not have single quotes, you must check to see if the attribute is expected to be a String attribute. If so, you should replace the pattern of "..." with "%{...}" so that the expression is still being evaluated.

Note that attributes such as disabled, maxlength, etc do not need the new syntax. That is becaus the syntax is only applied to attributes that are expected to be strings. This is very important to remember!

Finally, please note that this release is a preview release for the new syntax. The new syntax will not be finalized and turned on by default until 2.2.0. You are free to use it, but it is not guaranteed to be stable and may change in the coming releases.

Changelog

OpenSymphony JIRA (1 issues)

T	Key	Summary
↗	WW-581	JSP Tags should support better syntax

Release Notes - 2.1.5

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.5 Release Notes

Key Changes

- All UI tags now support the complete set of JavaScript event listeners now, such as onChange, onBlur, etc.
- ExecuteAndWaitInterceptor is easier to use
- Several important bug fixes for Velocity integration

Migration Notes

Version	Description	Old Code	New Code
2.1.4 and below	TLD updated - be sure you are using the latest webwork.tld file	N/A	N/A

Changelog

OpenSymphony JIRA (18 issues)

T	Key	Summary
↗	WW-666	ExecAndWaitInterceptor should put executing action on the stack
●	WW-663	VelocityResult doesn&apos;t initialize VelocityManager
↗	WW-660	xhtml&apos;s checkbox.vm vertical alignment
↗	WW-658	JSP Tags do not support onFocus, onBlur js handlers
●	WW-653	url taglib does not support &apos;page&apos; attribute but the webwork-example.war uses it all over the place.
↗	WW-651	IfTag does not convert to Boolean
●	WW-644	Xhtml generated by the ui tags is (still) invalid
↗	WW-632	Onclick for radiotag
●	WW-627	select.vm requires htmlEncode for name parameter
+	WW-626	Principal Interceptor
●	WW-614	Cannot set velocity macro autoreloading
●	WW-554	Bad value for IMAGES_URI in JasperReportsResult.java
●	WW-526	Velocity using include ignores character encoding
●	WW-479	getValueClassType() in ComboboxTag returns Boolean.class

↗	WW-425	Refactoring to decouple the UI Tag dependency on velocity
▣	WW-392	Update doubleselect tag
●	WW-223	Multipart & SaveDir
↗	WW-161	ServletDispatcher could be a lot simpler to extend

Release Notes - 2.1.6

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.6 Release Notes

Key Changes

This release includes a few bug fixes for the URL tag, a new base class to make type conversion easier, and better documentation. It also includes the latest XWork release: version 1.0.4

Changelog

OpenSymphony JIRA (4 issues)

T	Key	Summary
↗	WW-673	Create a WebWorkTypeConverter for extension
↗	WW-671	URLTag does not include parameters when value is specified
+	WW-664	Document WebFlow
↗	WW-585	ComboBoxTag should sublass TextFieldTag

Upgrading from 1.4

This page last changed on Mar 30, 2006 by scud.

Package changes

Webwork1.x was seperated into two projects, XWork and Webwork. From this, several classes have been moved to different package names.

- ActionSupport has moved from **webwork.ActionSupport** to **com.opensymphony.xwork.ActionSupport**
  - doExecute() no longer exists, override execute()
  - the methods addError and addErrorMessage are now addFieldError and addActionError respectively

Configuration changes

- **actions.xml/views.properties needs to be converted to xwork.xml**
  If you're using an actions.xml file to configure your webwork 1, you can use the attached XSLT to convert the actions.xml file to a vanilla xwork.xml file.
  To apply this XSLT, you'll need to do the following:
  Get a copy of the XSLT. You can find the latest version in CVS in webwork/src/etc/actions.xsl . Next, find yourself an XSLT rendering engine. Xalan is a good choice and can be found at http://xml.apache.org/xalan-j/index.html
  Finally, do the conversion.

```
java org.apache.xalan.xslt.Process -IN actions.xml -XSL actions.xsl -OUT xwork.xml
```

Remember that you'll need to Xalan libraries in your classpath to run the above command.
If you want to look at these pages directly in your browser, I recommend user Internet Explorer as it automagically formats XML documents reasonably. There one caveat though. WW1 had a way to shorten the declaration of actions by allowing you to specify a package prefix in webwork.properties file. Since this information is outside the actions.xml file, the XSLT is unable to take advantage of it. Consequently, you might need to edit the xwork.xml file to update the class names.

WebWork 1.x configuration used a pull paradigm to load action configurations when they are asked for, whereas WebWork2 builds the configuration up-front to make the configuration queryable. The webwork.MigrationConfiguration must therefore act as an adapter between these two paradigms. It does this by returning a custom RuntimeConfiguration which first tries the default XWork Configuration (which, by default, loads configuration information from a file named "xwork.xml" in the root of the classpath) and then attempts to load action configuration using the Configuration classes from WebWork 1.x. In this way, an application can be slowly converted over to WebWork2 while reusing the configuration and Actions from a WebWork 1.x application. One caveat in this is that your migrated application MUST be rebuilt against the WebWork2 and migration jar files, as the classloader will rightly recognize that the webwork.Action and webwork.ActionSupport in WebWork 1.x are not the same as those provided by the migration jar files. Other than that, it should be seamless (and let us know if it isn't).

If the webwork.MigrationRuntimeConfiguration does not find the action configuration using the RuntimeConfiguration from the supplied RuntimeConfiguration (which is acquried from the Xwork DefaultConfiguration and will load configurations from all of the sources configured for XWork / WebWork2), it will build an ActionConfiguration by instantiating an Action using the ActionFactories from WebWork 1.x. The ActionFactory stack used is a subset of the default ActionFactory stack used in WebWork 1.x:

```
factory = new JavaActionFactory();
        factory = new ScriptActionFactoryProxy(factory);
        factory = new XMLActionFactoryProxy(factory);
```

```
            factory = new PrefixActionFactoryProxy(factory);
            factory = new JspActionFactoryProxy(factory);
            factory = new CommandActionFactoryProxy(factory);
            factory = new AliasingActionFactoryProxy(factory);
            factory = new CommandActionFactoryProxy(factory);
            factory = new ContextActionFactoryProxy(factory);
```

Some of the ActionFactory classes have been left out as they are handled by Interceptors in WebWork2. If the Action instance is created (meaning that the configuration has been found in the webwork.properties or actions.xml files used by the WebWork 1.x configuration classes) a parameter Map is created by introspecting the Action instance. A Map is needed for results and, again, WebWork 1.x used a pull paradigm to find results when they were needed, so a LazyResultMap is created which extends HashMap and overrides get() to look up the Result configuration if it has not previously been loaded. If the result ends in the Action suffix (defaulting to ".action"), then a ChainingResult is created, otherwise a ServletDispatcherResult is created. Using the Action class of the instantiated Action, the Map of parameters introspected from the Action instance, and the LazyResultMap, a new ActionConfig is created. The ActionConfig is saved into a special Package, "webwork-migration", so that it will pick up the default Interceptor stack defined for that package. The "webwork-migration" package is defined in a webwork-migration.xml file which is included in the migration jar file and which is automatically added to the Xwork configuration providers when the MigrationConfiguration is used:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
    <include file="webwork-default.xml"/>
    <package name="webwork-migration" abstract="true" extends="webwork-default">
        <interceptors>
            <interceptor-stack name="migrationStack">
                <interceptor-ref name="timer"/>
                <interceptor-ref name="logger"/>
                <interceptor-ref name="chain"/>
                <interceptor-ref name="static-params"/>
                <interceptor-ref name="prepare"/>
                <interceptor-ref name="params"/>
                <interceptor-ref name="workflow"/>
            </interceptor-stack>
        </interceptors>
        <default-interceptor-ref name="migrationStack"/>
    </package>
</xwork>
```

Here we can see that a number of the functions previously performed by ActionFactories in WebWork 1.x are now replaced by Interceptors in WebWork2, including the parameters, the chaining, calling prepare(), and the workflow (which was formerly implemented in ActionSupport in WebWork 1.x).

By creating and saving the ActionConfig in the PackageConfig for the "webwork-migration" package, the ActionConfig for the migrated Action is available for future calls, obviating the need to re-parse the old configuration files and making the configuration for the Action available for querying via the configuration API for tools such as the configuration browser.

Tag Changes

The biggest change is the use of OGNL for accessing object properties. Properties are no longer accessed with a forward slash "/" but with a dot "." Also, rather than using ".." to traverse down the stack, we now use "[n]" where n is some positive number. Lastly, in WebWork 1.x one could access special named objects (the request scope attributes to be exact) by using "@foo", but now special variables are accessed using "#foo". However, it is important to note that "#foo" does NOT access the request attributes. "#foo" is merely a request to another object in the OgnlContext other than the root. See OGNL reference for more details.

Also see JSP Expression Language Comparison with WebWork 1.x for a table of the expression language changes.

property tag

The property tag is now only used to print out values from the stack. In WW1, it was also used to set a variable in the scope, and to push properties to the top of the stack. These functions are now performed by the set and push tags.

action tag

The action tag does not evaluate the body section any more and does not push the executed action onto the ValueStack. Instead, use the "id" attribute to assign a name to the action and reference it as "#id".

Examples

Lets enumerate some examples of differences between code snips using WW:WebWork and WW:WebWork.

- New JSP syntax

There are numerous changes in syntax. First of all there are new tags and secondly there is a new expression language. Here's a small example:

Webwork 1

```
<ww:property value="a/b">
  <ww:property value="foo" />
</ww:property>
```

Webwork 2

```
<ww:push value="a.b">
  <ww:property value="foo" />
</ww:push>
```

One can note that the "push" tag doesn't just push it pops too at the end of the tag. Surprise! Also note the "." instead of the "/" for traversing object properties.

- List errors posted by an Action
  Webwork 1

```
<webwork:if test="hasErrorMessages == true">
  ERROR:<br />
  <font color="red">
    <webwork:iterator value="errorMessages">
      <webwork:property/><br />
    </webwork:iterator>
  </font>
</webwork:if>
```

Webwork 2

```
<webwork:if test="hasErrors()">
  ERROR:<br />
  <font color="red">
    <webwork:iterator value="actionErrors">
      <webwork:property/><br />
    </webwork:iterator>
  </font>
</webwork:if>
```

Update your web.xml file

- If you're using Velocity for views, you'll need to make sure you have the following snippet. Specifically note that the <load-on-startup> tag is now required so that the servlet can initialize some important Velocity properties.

```
<servlet>
    <servlet-name>velocity</servlet-name>
<servlet-class>com.opensymphony.webwork.views.velocity.WebWorkVelocityServlet</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
```

- Set the property **webwork.velocity.configfile** in your _webwork.properties_. For example:

```
webwork.velocity.configfile=velocity.properties
```

WebWork will use this file to initialize the Velocity engine. The search path for the file is:

1. context root (web root)
2. WEB-INF/
3. classpath

- Additional Steps:

1. If you used the <ww:action taglib in 1.3... you used to refernece the java Action classname. In 2.x this reference is now the action name not the class. you will need to change all your old references in your view.

ResultException doesn't exist anymore

It might be possible to copy WW1's ResultException, and write an Interceptor that catches the ResultExceptions and add the result of getMessage() to the actionErrors of the executed Action and return ResultException.getResult().

Maybe it would be possible to include ResultException in WW2 too to make migration easier?!

DateFormatter doesn't exist anymore

It can be replaced by directly using `java.text.DateFormat`

addError(String, String) in webwork.action.ActionSupport has been removed

The new method to use is addFieldError(String, String).

addErrorMessage(String) in webwork.action.ActionSupport has been removed

The new method is now addActionError(String).

webwork.util.ValueStack has been removed

The ValueStack is com.opensymphony.xwork.util.OgnlValueStack

The old methods pushValue and popValue are renamed to simply push and
pop.

An instance of the ValueStack can be obtained by using ActionContext.getContext().getValueStack instead of the old ValueStack.getStack().

*Aware-Interfaces have been removed

Instead of implementing ServletRequestAware etc the
[Servlet]ActionContext.getXXX-methods can be used to obtain application-map, request, response etc.

CommandDriven interface removed

The CommandDriven interface is removed. It is not neccesary to implement a special interface when working with commands anymore. Use the method attribute in your action-Element in xwork.xml to tell xwork which method to invoke on your action.

isCommand(String) method has been removed

You can see which alias you're accessing by doing this:
ActionContext.getContext().getActionInvocation().getProxy().getActionName()

JSP Expression Language Comparison with WebWork 1.x

This page last changed on Mar 30, 2006 by scud.

Situation	Previous (WW-1.4)	Current (WW-2.1)
Referring to an object in the PageContext scope	@itemIdOrName	#attr[’itemIdOrName’]
Referring to an object in the Request scope	itemIdOrName	Same, but use #request[’itemIdOrName’] if nested in an iteration.
Referring to an object in the Session scope	@itemIdOrName	#session[’itemIdOrName’]
Referring to an object in the Application scope	@itemIdOrName	#application[’itemIdOrName’]
Property Setters	foo/bar translates to getFoo().setBar()	foo.bar translates to getFoo().setBar()
Property Getters	foo/bar translates to getFoo().getBar()	foo.bar translates to getFoo().getBar()
Boolean/boolean Property Getters	foo/bar translates to getFoo().getBar() if bar is java.lang.Boolean, if primitive bar translates to getFoo().isBar()	Same, except uses dot notation instead of a slash (i.e. foo.bar)
Collections as Properties	N/A	Collections (including arrays) are similar to other objects, except they allow indexing: foo.bar[indexOrKeyName] translates to getFoo().getBar().get(indexOrKeyName)
curly braces - {}, evaluates contents of braces first, and use the result as the property to then evaluate.	<webwork:property value="{’name’}"/> translates to getName() on the curent object.	No longer used.
Reference a static variable	@com.fully.qualified.class.name.TheActualClass@STATIC_ATTRIBUTE_NAME	Still in use.

Originally written by Jay Bose and sent to the mailing list

Upgrading from 2.0

This page last changed on Mar 30, 2006 by scud.

Upgrading from Webwork 2.0 is rather trivial. This version of webwork adds enhancements and bug fixes with hardly any configuration or syntax changes. Follow these two simple steps and you should be on your way with the latest and greatest from the OS crew.

1. Update/Replace your current binaries with the new binaries located in the distribution download under the `lib/core`. You may also want to grab any related jars from the `lib/optional` folder. Don't forget the webwork binary `webwork-2.1.jar` in the base directory of the distribution download. Review the Dependencies for Webwork.
2. Check out the Release Notes \- 2.1 to see if any of changes need to be applied to your code base.

Upgrading from 2.1

This page last changed on Mar 30, 2006 by scud.

Upgrading from 2.1 to 2.1.1 is very easy. Simply copy over the new webwork.jar file and make sure you are running all the correct Dependencies.

Upgrading from 2.1.1

This page last changed on Mar 30, 2006 by scud.

Upgrading from 2.1.1 to 2.1.2 is very easy. Simply copy over the new webwork.jar file and make sure you are running all the correct Dependencies - especially make sure you are using XWork 1.0.3.

Upgrading from 2.1.2

This page last changed on Mar 30, 2006 by scud.

Upgrading from 2.1.2 to 2.1.3 is very easy. Simply copy over the new webwork.jar. There have been no new dependencies.

Upgrading from 2.1.3

This page last changed on Mar 30, 2006 by scud.

Since this release is merely a preview release of the new tag syntax (see Release Notes \- 2.1.4) and no other changes have been made, simply copying over the new webwork jar file is all that is needed to upgrade. No other libraries have changed.

Upgrading from 2.1.4

This page last changed on Mar 30, 2006 by scud.

Upgrading from 2.1.4 is pretty easy as no new libraries have been introduced. However, the TLD (taglib definition file) has been modified to support more JavaScript events, such as onBlur, onChange, etc. Be sure to check that you are using the latest TLD. This is automatic if you point web.xml to /WEB-INF/webwork.jar.

Upgrading from 2.1.5

This page last changed on Mar 30, 2006 by scud.

Upgrading from 2.1.5 to 2.1.6 requires that you copy over the new webwork-2.1.6.jar and xwork-1.0.4.jar. Note that WebWork 2.1.6 includes a new version of XWork, version 1.0.4.

WebWork 2.1.7

This page last changed on Mar 30, 2006 by scud.

WebWork 2.1.7 release notes

Key changes

- XWork upgraded to 1.0.5
  - Using i18n, especially the ww:text tag, is now possible in SiteMesh decorators. See SiteMesh.
  - Issues where the ActionContext wasn't properly cleaned up (after ww:include and ww:action) have been resolved.
  - Configuration
    - If the action class is not specified, it now defaults to com.opensymphony.xwork.ActionSupport.
    - If the result name is not specified, it is assumed to be "success".
- Non-UI tags
  - ww:text and ww:url have an id attribute that, when specified, causes the tag not to print out the localized text but rather store the result in the ActionContext to be referenced later. See SiteMesh, URL tag, and Text tag.
- UI tags
  - ww:form has an added target attribute.
  - ww:form exposes "namespace" in the parameters Map in templates.
  - ww:form tag defaults the id and name attribute to the action name that it is submitting to.
  - Form elements default the id attribute to "[formId]_[elementName]".
  - Form elements default the "required" attribute to true if there is any validator associated with that field and the form's validate attribute is enabled.
  - ww:textarea has an added "wrap" attribute.
  - XHTML theme: added a parameter called "after" and that will add text to the right of the form element.
- Replaced old prototype client-side validation with XmlHttpRequest-based solution (STILL IN PROTOTYPE PHASE, works in XHTML theme only).
- Interceptors
  - ServletConfigInterceptor now checks for actions implementing PrincipalAware and adds a Principal that ties in to the HttpServletRequest's Principal.
  - Minor bug fixes in the ExecuteAndWaitInterceptor and FileUploadInterceptor.
- Results
  - JasperReportsResult and StreamResult both support the Content-disposition header.
  - JasperReports integration has been upgraded to 0.6.3.
  - WebWorkResultSupport has a simple conditionalParse() method making support for ${} notation in your results easier (it will parse only if the parse attribute is true).

Upgrade steps

1. Copy over the new webwork.jar and xwork.jar files.
2. If you are using JasperReports, you will need to upgrade to the latest JasperReports jars of 0.6.3 because the package name has changed.
3. Read the migration notes for anything that you should be aware of.

Migration notes

- If you were using the old client-side validation code, it is still possible to keep using it but you will likely need to modify form-close.vm and controlheader.vm in the XHTML theme and form.vm in the simple theme. We recommend supporting the new prototype as it is a lot easier to use and doesn't require special validators.

- It is possible, though very unlikely, that the default ids for the UI tags and the default name for the form tag might cause you some trouble. Please be aware of this change.

Changelog

OpenSymphony JIRA (26 issues)		
T	Key	Summary
●	WW-1067	Error undeploying webapps on Tomcat
↗	WW-706	Text tag and URL tag should have option to store contents to context
↗	WW-704	Integrate PrincipalInterceptor with ServletConfigInterceptor
↗	WW-703	Add wrap attribute to textarea
↗	WW-702	Required (*) should be on by default if a validator exists
↗	WW-701	UI tags should have default ids and names
●	WW-698	webwork.i18n.encoding does not get set by the ServletDispatcher
↗	WW-696	StreamResult should accept file option
●	WW-695	JSP Form Tags does not have a target attribute
●	WW-694	ExecuteAndWaitInterceptor can return null results
●	WW-693	FileUploadInterceptor don&apos;t check the allowed files&apos;s Enumeration whitch is null.
↗	WW-692	Webwork package of jasper reports packaged as dori.jasper. Latest release of Jasper Reports packaged as net.sf.jasperreports
↗	WW-691	JasperReports 0.6.3 support
↗	WW-687	Freemarker - add method buildUrl to FreemakerWebworkUtil
↗	WW-686	JasperReport result - Content-Disposition management
↗	WW-684	Default action class and result name.
●	WW-681	getText() doesn&apos;t work in Sitemesh filters
●	WW-677	ww:include replaces existing value stack with new one
●	WW-676	Freemarker Support:TemplatePath in web.xml has no effect
↗	WW-668	Externalise the JavaScript validation support from the JSP taglibs
●	WW-638	Freemarker result encoding error
↗	WW-637	textarea does not have maxlength attribute
●	WW-617	Stale Action Invocation left in Stack Context
●	WW-490	Is WW ignoring webwork.locale setting?
↗	WW-455	Select tag template does not work properly for Object like BigDecima

	WW-351	Forwarding unknown tag attributes

This page last changed on May 19, 2006 by scud.

WebWork 2.2 发行公告

主要变化

Productivity(生产力?) 提高

工具

- 全功能的WebFlow支持,包括JSP, FreeMarker, 和Velocity
- QuickStart: 简单嵌入式的application server可以快速体验

文档

- 改进的文档,包括每个拦截器的详细的信息
- 全新的示例程序: 现在的例子程序是一套仅对WebWork的最佳实践进行演示的教程, 而不是对每一个单独特性的演示.

增强的框架反馈

- 更好的,更多的智能错误报告
- "Developer" mode(开发模式) 只要可能就在线显示错误
- 一般的拦截器堆栈问题, 例如没有"input"结果的 validation+workflow , 以更明显的方式报告

其他

- 不再推荐 WebWork IoC container,推荐使用 Spring
- Spring 的内在支持
- 官方支持 wizards/workflows, 使用 Scope Interceptor 和 Continuations 的预览版
- 移除 WebWork 1.x 移植支持

用户界面改进

界面标签大力改进

- FreeMarker 现在是缺省的界面标签实现
- 重构的界面标签基类不在和JSP绑定
- 新的原生Velocity 和 FreeMarker 界面标签支持, 建立在新的标签基类之上
- 界面标签使用 "altSyntax" (2.1.4版本以来开始支持) 语法作为缺省的语法 (2.0 - 2.1 原有语法依然可用但是不推荐使用)
- 新的 Head 标签用来包含对应的theme的对应的CSS和JavaScript文件

Velocity 支持增强

- 支持升级到 velocity 1.4
- webwork.velocity.contexts 现在连接每个request的contexts , 例如 contexts 不需要线程安全了

AJAX 支持

- 正式支持使用DWR作为客户端校验
- 新的选项卡部件
- 内建对Dojo wdigets 的支持

Result 改变

- Velocity 和 FreeMarker Servlets 不再推荐使用,而推荐使用对应的result

其他

- 非常方便的调用不同的action及其方法, 使表单可以轻松使用多个按钮
- 开始进行JSR168整合

核心 API 变化

类型转换

- 支持 Maps, Sets, 和 Lists,甚至集合不为空时也支持
- Map 类型转换支持 keys 和 values
- 支持 Java 5的集合和enums(枚举)的泛型和标注

其他

- 改进的异常处理,包括异常到Result的映射(在xwork.xml里)
- 参数处理拦截器已经更新,可以让你包含或者排除一定的参数,这样就提供了一种安全的方法来保护你的数据不会被从web上被改变

迁移事项

WebWork 2.2 是2.0发布以来最具有标志性的发行版本. 有很多重要的变化,不推荐的项目,以及非常多的在你升级时需要注意的事项(如果你是刚开始使用,则无须注意). 请查看 WebWork 2.2 Migration Notes 了解更多信息.

修改日志

查看所有的修改记录,可以查看 完整修改记录

OpenSymphony JIRA (328 issues)		
T	Key	Summary
🔴	WW-1078	Broken links in Shopping-Cart
🔴	WW-1077	setting theme in page, context, or session scope has not effect
🟨	WW-1076	Move all Velocity templates to archive
🟨	WW-1075	url validator docs
🟨	WW-1074	AJAX docs screwed up

☒	WW-1073	control tags docs snippets are borken
●	WW-1072	select.ftl emptyOption evaluation
●	WW-1071	DoubleSelect&apos;s second select does not allow a predefined value to be selected
●	WW-1070	double select created option with key and value wrongly
↗	WW-1069	Ability to use freemarker map built-ins (?keys, ?values) as well as plain map methods (.keySet(), get(foo))
↗	WW-1065	TestCase for the Property component
☒	WW-1064	have a link in tags page to altsyntax page
✚	WW-1062	Add a OptionTransferSelect component
●	WW-1061	checkboxlist.ftl does not allow disabled parameter handling in WW2.2beta4
☒	WW-1059	Make the dojo configuration in ww:head hook into the i18n infrastructure
●	WW-1058	xhtml theme&apos;s form-validate.ftl onsubmit javascript is bad
●	WW-1057	DefaultActionMapper appending extra slash when namespace is "/"
↗	WW-1056	Change defaultStack order
↗	WW-1055	Warn when properties are null
☒	WW-1054	Include common build in project
☒	WW-1053	AJAX Validation Documentation
●	WW-1052	Always cleanup ActionContext
↗	WW-1051	Enable tag attribute description to be usable for both javadoc and tagdoclet
☒	WW-1050	Add DWR 1.1-beta2 jar to ivy-repository
●	WW-1049	ChainingInterceptor doesn&apos;t accept null of CompoundRoot element
●	WW-1048	RemoteCallUIBean uses UrlHelper to build url for href attribute breaks <ww:a /> tag
●	WW-1047	Fix broken locale files of jscalendar
●	WW-1046	DefaultActionMapper ignores the root ("/") namespace
↗	WW-1045	Include datepicker css in new ww:head tag
✚	WW-1044	Add a fielderror tag to display field errors if exist
●	WW-1043	Validation Errors with DWR (AJAX Tutorial)
●	WW-1041	Showcase - Edit Employee form broken
↗	WW-1036	Include datepicker css in new ww:head tag
✚	WW-1035	Add a actionerror tag that will display action error if they exist

↗	WW-1034	Datepicker uses wrong date format for en locales
↗	WW-1033	Generate tld attribute descriptions from property javadoc
☒	WW-1032	Document new head tag
●	WW-1031	Remote forms don&apos;t work with new dojo
↗	WW-1030	Loading Freemarker templates with the ActionContext&apos;s Locale
●	WW-1025	Showcase example: Spring throws strange exception while wiring action instance
●	WW-1024	dist build target is broken, webapps are not longer packaged
●	WW-1023	Item keys should use findValue
●	WW-1022	A tag doesn&apos;t work with params
●	WW-1021	VelocityTools 1.2 ToolboxManager implementation causes NPE in VelocityManager
↗	WW-1020	Consolidate ww:a and ww:href tag into ww:a
✚	WW-1019	Pico/nano integration
☒	WW-1017	Update DOJO to release 0.20
↗	WW-1016	Merge Quick Start Guide into Getting Started document
●	WW-1015	Cannot create Portlet instance com.opensymphony.webwork.portlet.WebWorkPortlet for Portlet Application webwork
●	WW-1014	i18n reloading in Tomcat
●	WW-1013	Typo!
●	WW-1012	tabbedpanel-close.ftl is broken due to iterator refactoring
●	WW-1010	AbstractListTag no longer allows nulls for the list attribute
☒	WW-1009	AJAX tutorial broken
●	WW-1008	Iterator tag throwing NullPointer
☒	WW-1007	SiteMesh docs out of date
☒	WW-1006	Java 5 support
☒	WW-1005	Action chaining docs
☒	WW-1004	OGNL docs need new examples
☒	WW-1002	I18n docs out of date
☒	WW-1001	Client side validation
☒	WW-1000	Validation examples outdated
☒	WW-999	Make all result types based on snippets
☒	WW-998	Document Exception Handling
☒	WW-997	document for each validators type its usage, parameter and example using snippet
●	WW-996	Type conversion not working for map
☒	WW-995	Please update documentation for the FileUploadInterceptor -

		allowedTypes parameter.
●	WW-994	Component based IteratorTag never prints out is body
⊠	WW-993	Add Exception Mappings to Config Browser
●	WW-991	FilterDispatcher setup and cleanup non-webwork request
●	WW-990	is extremely unhelpful for new people a exception thrown message
●	WW-989	WW Action Tag does not go through ActionMapper
↗	WW-987	org.opensymphony used in text constants instead of com.opensymphony
↗	WW-986	Re-add uri declaration in WW 2.2 tld
●	WW-985	New FM templates don&apos;t support booleans very well
⊠	WW-984	Please define the scope interceptor in webwork-default.xml
●	WW-983	option values are locale specific formatted - should only be HTML escaped.
●	WW-982	Quickstart broken on OS X
↗	WW-981	Calls made to Configuration instances are not centralized constants.
⊠	WW-979	Verify on WebLogic 9.0
●	WW-977	Input and Output streams not closed in StreamResult
●	WW-976	WW Portlets can&apos;t be deployed in Liferay or JBoss Portal
●	WW-975	Snippet macro doing weird things with text tag
●	WW-974	set useAltSyntax problem
↗	WW-973	Make datepicker locale aware
●	WW-972	ww:property tag does not recognize alt-syntax
↗	WW-971	Setup XDoclet to build tag documentation and tld from component sources
●	WW-969	Fix Subset Tag
●	WW-968	Fix Append Tag
●	WW-967	Fix Merge Tag
●	WW-966	i18n issue, locale is randomly switched
●	WW-965	Fix WW Generator Tag
↗	WW-964	Support for JasperReports 1.1.0
↗	WW-963	Add overridable publishException method to ExceptionMappingInterceptor
●	WW-961	chainStack defined twice in webwork-defaults.xml
●	WW-960	Action tag does not do include properly
●	WW-957	TabbedPaneTag doesn&apos;t have openTemplate setter
●	WW-956	UIBean NPE with ww.submit tag

↗	WW-955	Switch to using [and] in Freemarker templates
●	WW-954	Freemarker does not handle map correctly, cant lookup value
●	WW-953	Very bad performance using new WW 2.2 tags
●	WW-952	Superflous logging with config-browser
●	WW-951	Config browser problem
●	WW-950	config-browser showConfig.action not work
↗	WW-949	UI Form element should support theme attribute
↗	WW-948	Make build from distribution package self-contained
●	WW-947	Sample webapps build process is broken in the distribution
⊠	WW-946	File interceptor should allow all files unless otherwise specified
●	WW-944	Disabled namespace attribute when not using AJAX validation
●	WW-943	AJAX validation and devMode don&apos;t play nice
●	WW-941	Cleanup and SiteMesh problems
●	WW-940	Dates cause problem with UI tags in FM
↗	WW-939	Clean-up the Release Notes
●	WW-938	WW:Sort Tag is not working
●	WW-937	theme css xhtml is missing form-close.ftl
●	WW-936	problem with ww:form tag
+	WW-933	Add a page to test all UI components
⊠	WW-931	Add dojo Color-Chooser functionality.
●	WW-930	ClassCastException in WW2 label tag rendering when value expression evaluates to a non-String type
●	WW-929	Freemarker rendering of input tags should not apply locale formatting
●	WW-928	Seems like download page gives wrong file of ww 2.2 beta 2
●	WW-927	ww 2.2 beta binaries is available for downloading, but xwork 1.1 is not
●	WW-926	Type conversion fails with ModelDriven actions
⊠	WW-925	Document the config-browser in Related Tools
⊠	WW-923	Fix up validation documentation
●	WW-921	Defect in com.opensymphony.webwork.views.util.ResourceUtil
●	WW-920	Missing attribute for ww:form tag
●	WW-919	Bug in DefaultActionMapper (could be Weblogic specific)
●	WW-918	Setting webwork locale or Action locale has no effect on JasperReports result
●	WW-917	HttpServletRequest locale/encoding

		problem
☒	WW-916	Complete Architecture section of Documentation
☒	WW-915	Complete Introduction section of Documentation
➕	WW-914	Create new ww:errors tag
➕	WW-913	Alt Syntax Migration - Page Specific altSyntax Change
➕	WW-912	ww:a support for preInvokeJS
☒	WW-911	Fix velocity code bug
☒	WW-910	Update the IOC section to reflect Spring intergration.
●	WW-909	Document FreeMarker simple map change
●	WW-907	ww:a does not support nested param tags
▣	WW-905	Update Scope Interceptor Documentation
▣	WW-904	Update Execute and Wait Interceptor Documentation
▣	WW-903	Update File Upload Interceptor Documentation
▣	WW-902	Update Chaining Interceptor Documentation
▣	WW-901	Complete Conversion Error Interceptor Documentation
▣	WW-900	Complete Prepare Interceptor Documentation
▣	WW-899	Complete Servlet Config Interceptor Documentation
▣	WW-898	Complete Workflow Interceptor Documentation
▣	WW-897	Complete Validation Interceptor Documentation
▣	WW-896	Complete Token Session Interceptor Documentation
▣	WW-895	Complete Token Interceptor Documentation
▣	WW-894	Complete Component Interceptor Documentation
▣	WW-893	Complete Model Driven Interceptor Documentation
▣	WW-892	Complete Parameters Interceptor Documentation
▣	WW-891	Complete Static Parameters Interceptor Documentation
▣	WW-890	Complete I18n Interceptor Documentation
☒	WW-889	Complete Interceptor Documentation
☒	WW-887	Clean up FAQ
☒	WW-885	Document how validation picks up a default validators.xml now
☒	WW-884	Document new type conversion/collection stuff
☒	WW-883	Document general FM result integration (very basics)
☒	WW-882	Document general jsp result integration (very basics)
☒	WW-881	Document general velocity result integration (very basics)

☒	WW-880	Rename Prototype to QuickStart
☒	WW-879	Rename WebFlow to SiteGraph
☒	WW-878	Document ivy build process and how dependencies are handled
☒	WW-877	Start new, clean "best practices" section
☒	WW-876	Document how to launch "WebFlow" (SiteGraph?)
☒	WW-875	Document changes to the Session Map impl
☒	WW-873	Document how static content is served
☒	WW-872	Document date picker tag
☒	WW-871	Document altSyntax and TagSyntax
☒	WW-870	Document JSP-only tags
☒	WW-869	Document JSP tag syntax (like Velocity and FM)
☒	WW-868	Document each generic tag
☒	WW-867	Document new webwork.properties settings
☒	WW-866	Document ActionMapper feature
☒	WW-865	Document ParameterNameAware feature
☒	WW-864	Document new redirect-action result type
☒	WW-863	Document special button names (action:, etc)
☒	WW-862	Document themes and new theme extension feature
☒	WW-861	Pure client side validation
☒	WW-860	Document deleted/deprecated items
☒	WW-859	Document AJAX features
☒	WW-858	Document prototype/launcher info
☒	WW-857	Document execAndWait changes
☒	WW-856	Document file upload interceptor/file upload
☒	WW-855	Document i18n interceptor
☒	WW-854	Document continuations
☒	WW-853	Document XWork IOC deprecation
●	WW-851	webwork.components.Form small bug:
●	WW-850	taglib.tld form tag missing afterLoading attrobite
●	WW-849	reloadingText attribute not on PanelTag
●	WW-848	example webapp starter throws NullPointerException in com.acme.CreatePerson.execute( )
●	WW-847	WW2.2 Tag Syntax Issues
●	WW-846	css xhtml validation.js addError IE bug fix
●	WW-845	css xhtml/submit.ftl remove controlfooter include
●	WW-844	JspTaglibs in freemarker isn't

		working
●	WW-842	datepicker always set the value in the last field
●	WW-841	Illegal flush with URL Tag and Sitemesh Tags
●	WW-840	Shopping cart add to cart not working
➕	WW-839	css xhtml theme
●	WW-836	DispatcherUtils not JDK 1.3 compliant
↗	WW-835	FileUploadInterceptor should use a deny-all policy instead of both a allow and disallow.
●	WW-834	Freemarker CallbackWriter does not properly write out content if usesBody() is true
●	WW-833	DWRValidator not populating entire context
●	WW-832	Shoping-Cart Example Has Several Issues
●	WW-831	Incorrect wrap attribute of the TextArea UI component
●	WW-829	VelocityResult does not always call OutputStreamWriter flush() method
●	WW-828	maxlength is broken
●	WW-825	The onsubmit attribute in the form tag is missing
●	WW-824	redundant ServletDispatcher error handling complicates filters
●	WW-823	Displaytag still doesn&apos;t work
●	WW-821	FileUploadInterceptor Error When not set parameter,and should save message in properties
☒	WW-820	Reconcile dojo with the newest release
●	WW-819	Orionserver filter request doesn&apos;t contain action name in getServletPath()
●	WW-817	Sitemesh decorator not finding actions when using FilterDispatcher
➕	WW-815	Ability to Specify ObjectFactory
↗	WW-813	Make ExecuteAndWaitInterceptor easier to extend
●	WW-812	JasperReports: support for delimeter
●	WW-811	StreamResult doesn&apos;t close InputStream
●	WW-810	Iterator Sort tag fails
↗	WW-806	Better logging of parameters interceptor
●	WW-804	Not clear where ww2 looks for webwork.velocity.configfile
●	WW-800	ACTION NAME is never put on the stack
●	WW-798	JakartaMultiPartRequest does not set the header encoding, so the filename can not be parsed correctly when using a UTF-8 encoding in request.
●	WW-792	ActionTag does not replace old stack on exception

●	WW-791	Configuration browser does not work with velocity 1.4
●	WW-790	WebWork does not refresh ressource bundle when told to do so
●	WW-787	support for onclick scripting event missing in checkboxlist tag
↗	WW-786	Add contains to ComponentTag
●	WW-784	ww:property does not escape HTML by default
↗	WW-783	Checkboxlist.vm in xhtml should point to Checkboxlist.vm under simple template dir
●	WW-781	Velocity contexts are shared between threads
●	WW-780	Manifest problem with dependency commons-logging
●	WW-779	WebWorkTagSupport.translateVariables() doesn&apos;t work with lists or maps
●	WW-778	WW2: IfTag does not always return true or false
●	WW-776	MultipartRequestWrapper swallows errors coming from a parser
●	WW-775	ww:if/else expression bug
●	WW-774	WW framework creates extraneous Session objects
●	WW-773	UrlHelper SHOULD use &amp; entity instead of & to add GET parameters
●	WW-768	The label.vm template uses attribute "id" instead of "for"
●	WW-767	2.1.7 AbstractListTag throw NullPointerException in method evaluateExtraParams
↗	WW-763	True velocity macros for UI tags
●	WW-761	FileUploadInterceptor does not use parameters defined in JavaDocs.
●	WW-760	white space issue with webwork.custom.i18n.resources webwork.properties property
●	WW-759	LogFactory.releaseAll() should be called at application end
↗	WW-758	Use Introspector.flushCashes() to deal with memory leaks on application reload
●	WW-756	broken link in wikidoc
➕	WW-755	XMLHttpRequest component to replace DOM node with new HTML
●	WW-754	Infinite Loop in XSLTResult&apos;s DOM adapter
●	WW-753	Website 404s
●	WW-751	xslt view support needs to implement additional methods to support 1.5
●	WW-748	StreamResult should be close inputStream
●	WW-747	URLHelper class does not use charset for encoding
●	WW-745	no scripting attributes on the submit tag
●	WW-744	Tutorial needs to have latest jar

●	WW-743	execAndWait Interceptor Override Default Action Method
●	WW-742	wrap attribute in textarea.vm in &apos;simple&apos; template
↗	WW-740	UI tag checkboxlist should have html scripting events attribute
●	WW-739	Action tag TLD missing ignoreContextParams
↗	WW-738	Use ObjectFactory in velocityManager for setting webwork.velocity.contexts
↗	WW-737	JasperReports examples in the webwork-example app not working
●	WW-736	XmlHttpRequest validation breaks on IE
●	WW-735	$webwork.evaluate() throws NPE
●	WW-730	2.1.7 text taglib is not compatible with old version
●	WW-729	UrlTag always includes parameters unless includeParams is explicitly set to false
●	WW-725	The include tag doesn&apos;t cleanup after itself
●	WW-721	<ww:subset/> problem
↗	WW-720	Add content-length to StreamResult
↗	WW-719	Session object not auto-shared with Velocity context
●	WW-717	The "label" tags is missing the "for" attribute
●	WW-713	The <ww:url> tag does not correctly construct the URL when the scheme attribute is set and the scheme changes.
↗	WW-712	Update config browser to work with the new syntax
↗	WW-709	Allow specifying a text for empty option on the select tag
●	WW-708	AbstractListTag classes throw NPE for null value
●	WW-707	Velocity templates in UTF-8 still do not work
●	WW-700	Validation lacking logging of missing field.
+	WW-697	Regex Field Validator
●	WW-682	change webwork.ui.templateDir=otherDir, tag class, doEndTag: Unable to find resource &apos;/template/simple/scripting-events.vm&apos; error.
+	WW-679	The "Component" UI tag can&apos;t be used as part of a Velocity #bodytag directive
●	WW-672	URLTag does not reset between invocations
●	WW-654	Error occurs with Page Context when having request dispatcher code in the Action code
●	WW-650	CoolUriServletDispatcher throws StringIndexOutOfBoundsException
●	WW-648	Client-side validation doesn&apos;t

		allow cancelling
↗	WW-636	Improve URL-Tag to support value stack and namespaces
●	WW-621	Lesson 2 web.xml refers to webwork-2.1.jar file
↗	WW-609	Velocity context available in Sitemesh 2.1 and WebWork use different keys for the request and response
+	WW-603	Create a NumericFieldValidator using JavaScript
●	WW-596	IllegalArgumentException when setting an indexed property
●	WW-591	Wrong output when using nesting #bodytags (with a custom tag)
↗	WW-589	Printer-friendly of all documentatioon
●	WW-588	Unexpected behaviour of SessionMap.clear()
●	WW-572	Lesson 2 incorrectly references the xwork-validator-1.0.dtd for validators.xml
●	WW-570	Table tag parameters and sorting
+	WW-558	Exception Handler Interceptor and Exception Action
●	WW-557	Generator Tag doesn&apos;t work properly
↗	WW-556	Allow ComponentManager scopes to be independant
+	WW-540	Add a ServletFilter to clean up the ActionContext, allowing Sitemesh decorators to use the ActionContext with the taglib
●	WW-536	Lost the session on OC4J 9.0.4 using WebWork2
●	WW-533	Add support for Velocity 1.4 - Changes WebWork directives
↗	WW-528	Switch to using VelocityViewServlet from VelocityTools
●	WW-520	Wrong &amp; management in XML based output (WML and XHTML)
●	WW-513	IOC Application and Session scopes do not work in Tomcat 5
●	WW-507	"command" parameter isn&apos;t honored for command driven actions
●	WW-505	Re-opened WW-430 (double action tag bug)
↗	WW-491	Spring/Xwork Action configuration
●	WW-489	Setting of pageContext in directive (tag)
↗	WW-477	WebWork needs a simple "Hello World" demo app
+	WW-463	ww:error tag
↗	WW-452	Pell Multipart library dies if content type is too long
●	WW-447	CoolUriServletDispatcher errors related to getServletPath()
+	WW-439	Simple wizard framework
●	WW-436	ParamTag inconsistently evaluates the &apos;name&apos; attribute
↗	WW-416	Reload resourcebundle from

		filesystem during runtime
🔴	WW-405	Escaping text in form fields from velocity tags not working.
🔴	WW-395	urltag doesnt show the exact url used
↗	WW-394	ServletDispatcher to use requestURI instead of servletPath
➕	WW-341	ScopeInterceptor
➕	WW-337	Text formatting support from tag library
➕	WW-329	Date picker component
➕	WW-304	client-side validations based on validation framework declared validations
🔴	WW-291	Too sophisticated exception handling in servlet dispatcher
↗	WW-270	Component UI tag should allow body content manipulation
🔴	WW-220	Velocity template for ui:textarea needs improvement
➕	WW-159	ServletRedirectResult should be aware of context paths
➕	WW-157	Better JSTL Support
☒	WW-16	More velocity macros, etc
➕	WW-6	Support for Portlet API

WebWork 2.2 Migration Notes

This page last changed on May 19, 2006 by scud.

此文档涵盖了一步一步的指导,指导你从2.1.x升级到WebWork 2.2,同时也是一个主要的变化的指南的列表.

升级指导

1. 获取 最新的 2.2 发布版本
2. 检查 依赖库 看看需要哪些必须的库.一个需要注意的改变就是对Rife-Continuations的依赖. 点击各个tab了解不同使用情况下的依赖库.例如如果你使用FreeMarker,就点击那个tab来查看需要哪些依赖.注意如果你私用JSP标签,缺省情况下你就是使用FreeMarker作为UI组件的模板的.
3. 检查下面的 **单项改变** 部分来查看是否有一些影响你的代码的改变
4. 更新到使用 FilterDispatcher 来代替 ServletDispatcher. 检查 web.xml 2.1.x compatibility 页面查看一些兼容性的讨论, 以及查看 web.xml 来了解哪些东西需要放到 web.xml 文件里.

过时/废弃(不建议使用)的项目

- ServletDispatcher 已经被废弃了, 如果可能请使用FilterDispatcher. 浏览 web.xml 2.1.x compatibility 获取更多关于当你切换到FilterDispatcher时可能潜在的问题的更多信息.
- Velocity 和 FreeMarker servlet 已经不在被支持了.我们极力地推荐你不要使用这些servlet,而是直接使用 FreeMarker Result 或者 Velocity Result.
- 如果你在Velocity里面使用JSP标签,这不会在被支持了并且很快被移除. 你可以使用web.xml 2.1.x compatibility 里面所说的步骤来让它正常工作,但是我们极力推荐使用新的WebWork提供的原生的Velocity标签.
- 表格(table)的标签已经被确定废弃了.在将来如果更多的时间可以投入的话我们可能不在废弃它,但是我们推荐你寻找其他替换的方式,例如Display Tag.
- 当你使用FilterDispatcher时所有对包含action的支持(使用 include 标签或者 jsp:include) 不在有效. 我们推荐你使用 action 标签来代替.
- cos和pell文件上传解析器不在被积极维护,会被很快移除.我们强烈建议你使用Jakarta的文件上传解析器,也就是缺省的解析器.

被删除的项目

- 所有的 VoiceXML 标签已经从WebWork里移除.
- 基于Velocity的 Tags 已经被移除. 如果你使用或者扩展了这些标签(一般是高级用户),你可以从webwork的jar里面的/template/archive里面复制它们.

单项改变

版本	描述	老的代码	新的代码
2.1.x	如果你实现了你自己的 ObjectFactory 或者 ActionInvocation 类, 你会注意到有一些小的变化,对于build*方法可以使用一个 "extraContext" Map.这允许,例如在对象创建时访问Session map,甚至是在 ActionContext ThreadLocal 被设置之前.	ObjectFactory.getObjectFactory().buildBean(clazz);	ObjectFactory.getObjectFactory().buildB extraContext);

2.0+	如果你使用WebWork的基类来构建模板化的标签,你应该需要重构UI标签,并使用通用的 Component 类作为模板的后盾.现在标签都使用了这些 Component 类 ,Velocity和FreeMarker也是如此. 这允许 Velocity 和 FreeMarker 直接使用相同的UI组件,不再需要自称具有一个JSP页面.但是这也意味着需要重构你的自定义标签来使用新的API.	...your code..	查看2.2源码里已经存在的UI标签
2.1.x	如果你还 没有 使用 Alt Syntax, 它现在缺省是启用的了. 你现在不是选择升级就是改变 Tag Syntax	<ww:url value="'http://www.yahoo.com'"/>	<ww:url value="http://www.yahoo.com"/>
2.1.x	如果你使用了FreeMarker并且你的代码在collection和map上使用了psuedo 属性, 你需要修改代码来用调用方法的方式替代.	${parameters?size} / ${parameters.size?html}	${parameters.size()} / ${parameters.get("size")?html}
2.1.x	defaultStack 已经被重命名为 basicStack.	<interceptor-ref name="defaultStack"/>	<interceptor-ref name="basicStack"/>
2.1.x	completeStack 被重命名为 defaultStack.	<interceptor-ref name="completeStack"/>	<interceptor-ref name="defaultStack"/>
2.1.x	defaultStack (也就是以前的completeStack) 现在在 webwork-default.xml里是缺省的拦截器.另外,这个stack还配置了 Workflow Interceptor 和 Validation Interceptor 在遇到方法名为 input, back, 或者 cancel 的方法不在运行.	N/A	N/A
2.1.x	component 拦截器已经被废弃 (以及所有的 WebWork IOC 特性) 并且被从 basicStack 和 completeStack里移除. 如果你希望使用这些废弃的特性你需要手动把它添加回来.	N/A	N/A
2.0+	include 标签的 page 属性自从1.x就被废弃现在从2.2里面移除了. 请使用value属性.	<ww:include page="..."/>	<ww:include value="..."/>
2.0+	text 标签的 value0, value1, value2, 和 value3 属性自从1.x就被废弃了现在已经从2.2里移除了.请使用param标签代替.	<ww:text value0="..."""/>	<ww:text><ww:param value="someValue">...</ww:param></ww:t
2.0+	session map wrapper (在ActionContext里建立的) 已经改变了不在为每个请求创建session. 如果你的应用程序依赖session会被自	N/A	N/A

	动创建,WebWork 2.2已经不在那样做了.作为替代,你必须自己创建session或者当把一个数据放到session Map里时session会被创建.		
2.0+	VUI 标签已经从WebWork里移除.它们在4年中没有被积极地使用而且在社区中没有被使用.	N/A	N/A
2.0+	WebWork的TLD的URI已经从webwork 改为 /webwork. 如果你已经使用了在webwork.jar里大包的TLD,你必须在你的JSP里面调整URI.	<%@ taglib uri="webwork" prefix="ww" %>	<%@ taglib uri="/webwork" prefix="ww" %>
2.0+	缺省的编码已经从 ISO-8859-1 改为 UTF-8. 如果你希望继续使用 ISO-8859-1, 你必须修改你的 webwork.properties.	N/A	webwork.i18n.encoding=ISO-8859-1

WebWork 2.2.1

This page last changed on Apr 05, 2006 by scud.

WebWork 2.2.1 发行公告

主要变化

Portlet 集成

- JSR-168 portlet 集成支持 支持哪些portal Server
- 改进的 webwork-portlet 示例程序
- 为各个portal server添加部属描述文件

校验

- 改进 客户端校验

UI 标签

- 新的 Tree 组件
- 新的 UpDown Select 组件 updownselect
- 修正了 Debug 标签

工具

- Quickstart 改进
- 通过pom.xml支持Maven

其他

- 改进的 JSTL 支持
- showcase里面包含了更多例子
- 在starter webapp里面的简单快速上手例子
- 在webapps/build.xml里面的'ant new' 任务来创建你的项目的结构为你的web应用程序创建一个webapp项目框架

迁移事项

WebWork 2.2.1 是一个月前发布的2.2版本的错误修正版本.
如果你已经使用了前一个版本的portlet集成代码,你需要检查portlet-webapp这个例子.

在1.1版本中,xwork.dtd里面的DTD public ID是不正确的.确认你的xwork.xml的DOCTYPE定义看起来是这样的:

对于 xwork 1.1:

```
<!DOCTYPE xwork PUBLIC
        "-//OpenSymphony Group//XWork 1.1//EN"
        "http://www.opensymphony.com/xwork/xwork-1.1.dtd">
```

对于 xwork 1.1.1:

```
<!DOCTYPE xwork PUBLIC
        "-//OpenSymphony Group//XWork 1.1.1//EN"
        "http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">
```

修改记录

要查看完整的修改记录,请查看complete changelog

OpenSymphony JIRA (68 issues)

T	Key	Summary
●	WW-1203	wrong tag for freemarker result (ww.optiontransferselect, ww.actionerror, ww.actionmessage)
☒	WW-1143	Create blank webapp application with an ant target to create a getting started webapp
☒	WW-1142	Move starter webapp samples to showcase webapp
☒	WW-1141	Update ivy for 2.2.1 release
●	WW-1140	Version of Xwork DTD
☒	WW-1139	Document advanced features of Collection and Map type conversion
●	WW-1138	Included XWork librarys ....xwork.ObjectFactory.getClassInstance calls ....util.ClassLoaderUtil which actually exists with an s at the end
●	WW-1137	Dist target does not copy osbuild.xml to dist dir
+	WW-1136	Add a UpDownSelect component
●	WW-1134	Extension should not be limited for action mapper
●	WW-1132	webwork looking for /template/xhtml/a-close.ftl
↗	WW-1130	ability to programmatically set velocity properties
●	WW-1129	Debug tag not exposed to VelocityManager, DebugDirective missing
●	WW-1128	Sitemesh Velocity Integration renders uneval&apos;d templates
●	WW-1127	DevMode impacts pageflow logic
●	WW-1126	Quickstart won&apos;t work with Java 5 code
↗	WW-1125	optiontransferselect tag should add its respective entries when the form it is included is submitted
↗	WW-1124	optiontrasferselect tag uses capitals for html tag.
●	WW-1121	Freemarker error if "name" parameter is missing
↗	WW-1120	Allow setting templateSuffix on a per Page basis (With patch and

		UntiTests)
🟨	WW-1118	Improve portlet sample webapp
🔴	WW-1117	int validator in javascript throws exception if only min or max are set
🔴	WW-1116	ww:select tag works only with string variables
🟨	WW-1115	Refactor portlet implementation
▣	WW-1114	Stack overflow bug fix implemented incorrectly?
🔴	WW-1112	XML Field Validation Required not working on Integer or Long values
🟨	WW-1111	Debug component should be moved into c.o.w.components package
🔴	WW-1110	Debug Tag not working with FTL
🔴	WW-1109	the javascript in the DHTML response from remove DIV has not yet be evaluated
🔴	WW-1108	ognl.OgnlException: templateDir [java.lang.StackOverflowError]
➕	WW-1107	Client-side javascript for basic validators
🔴	WW-1106	More Config Browser issues
🔴	WW-1105	javadoc typo in TokenInterceptor
🟨	WW-1104	UITag tests need improvements
🔴	WW-1103	jsessionid params appended to css and js includes
🔴	WW-1102	Multipart requests are not cleaned up correctly
🔴	WW-1101	jsp view cannot generate onsubmit html tag correctly.
🔴	WW-1099	ScopeInterceptor has hardcoded dependencies on HttpServletRequest, etc
🟨	WW-1098	Maven Repository at ibiblio does not contain required dependencies for WW/XW
🔴	WW-1096	datapicker tag cannot display calendar correctly when browser&apos;s locale is zh_CN.
↗	WW-1093	Method Specification for Validation
🔴	WW-1092	Fix validation regexps for url and email in xhtml theme
🔴	WW-1091	WebWorkModels attribute names should be named as in TLD
🔴	WW-1090	debug.ftl template is broken
🟨	WW-1089	Update ivy descriptor for 2.2.1
🔴	WW-1088	the prefixs in DefaultActionMapper will cause ParametersInterceptor problem(OGNL parse error)
↗	WW-1087	make includeParams of URL don&apos;t override explicit params
🔴	WW-1086	Various 2.2 issues from Eugene
🔴	WW-1085	Datepicker i18n related javascript errors
🔴	WW-1084	quickstart CRUD skill example attempt to save/update result in

		error 500
🔴	WW-1083	Stack overflow in <ww:form> tag getTheme(), getAncestor(), when 2 forms on a page
🔴	WW-1082	ServletRedirectResult computes wrong URL when the root namespace is used
🔴	WW-1081	Some error for template\xhtml\form-close-validate.ftl
🔴	WW-1080	JSTL support not functioning correctly
🟨	WW-1079	Test portlet integration against the eXo platform
➕	WW-1068	Property "title" to jsp tags
↗	WW-1060	add tr attribute to ui tag component
↗	WW-1027	ValidationInterceptor and DefaultWorkflowInterceptor Improvements for forwarding to another action when meeting validation error
🔴	WW-958	Config-Browser don&apos;t respect webwork.action.extension = jspx
🟨	WW-908	Client side validation?
🔴	WW-906	href class has issues w/ altSyntax == false
🟨	WW-874	Document Portlet stuff
➕	WW-838	Create a new ww:css tag
↗	WW-830	PROTOTYPE: Add configuration options for flexibility
🔴	WW-818	URLEncoder.encode,ignore the charset
🔴	WW-795	Can&apos;t override default webwork messages
🔴	WW-793	Message with key "webwork.internal.invalid.token" is never used
↗	WW-688	Refactor TokenInterceptor and TokenHelper (and affected classes)

Projects Using WebWork

This page last changed on Mar 30, 2006 by scud.

- Atlassian Confluence - Commercial Wiki and knowledge management system using WebWork 2.0, Hibernate, Spring, and Velocity
- Jive Software - With over 1100 customers, Jive Software's Forums and Knowledge Base products both use WebWork 2.1+ and are some of the largest deployments of WebWork
- Midwinter - an open source rapid web application develop system using WebWork 2.0, Hibernate, Spring, and Velocity

- Chemist Australia - Online Pharmacy

- DriveNow - last minute Autralian car rentals

- OpenReports - an open source web based reporting application that uses WebWork 2.0, Velocity, and Hibernate

- eSage Group is a consulting company that uses it for all their client engagements. Additionally, its used for their internal systems

- Filmweb - Polish Film Portal

- TeraMEDICA - WebWork is used in TeraMEDICA's commercial TI2m product, which performs intelligent image management for the healthcare enterprise. Specifically, WebWork is a key component of the system's management interface.

- EBIA COBRA and 401K Benefits Site provides law reviews for employee benefits like COBRA and 401K. The site moved from all struts to a current architecture of about 50% WebWork and 50% Struts. We are trying to move it all over to WebWork. This site is also a great example of porting from Tiles to SiteMesh.

- Valtira Rolecall - Identity Management Framework that provides single sign-on for J2EE and .NET web applications.

- Orange Blossom Indian River Citrus- Retail site of shipper of fresh Florida (USA) citrus.

- JavaEye Reporting Tool - An open-source, web-based database reporting tool. It allows you to create reports without any programming (though you'll need SQL knowledge). It's a lightweight reporting environment, the report can be created to quickly share information via web.

- Dating-site - Commercial dating site (BE, dutch) ww2.1.7, hibernate

- No Fluff, Just Stuff - Java/Open Source Conferences delivered locally.

- Agility Issue and Bug Tracker - Bug tracking software that makes heavy use of WW's Ajax Framework. Take a look at the dashboard for great examples of what can be done with WW.

- JavaBB - phpBB like forum system. Written and used at Javafree brazillian community.

- CRI - A pharmecutical marketing compliance application.

- Logicd.com - A logistics, shipping management, tracking, supply-chain management application.

- Green Array - Green Array empowers managers at every level, in every organization. Green Array's web-hosted software solution is a simple, fast, scalable way to improve visibility, collaboration and to accelerate team performance. Integrated SWT client and WebWork 2.2 html dashboard with Hibernate persistence.

- Medihome Journal - A dutch/french site with free medical streaming videos.

QuickStart

This page last changed on Mar 30, 2006 by scud.

WebWork provides a quick way to get started called QuickStart. QuickStart is essentially a combination of a few technologies and some general conventions for developing web applications. What it lets you do is write applications without the need to even compile your sources, let alone have to deploy and redeploy them after every change. Instead, you can now develop your web applications just like if you were writing perl or PHP - on the fly and as quickly as you can think.

How to Use It

QuickStart is included in the WebWork distribution and can be launched by simply running **java -jar webwork.jar quickstart:mywebapp**. At this point you can access http://localhost:8080/mywebapp and begin developing your application. **At this time, QuickStart requires Java 1.5.**

Is Port 8080 really free for use?

If you face problems while starting applications via quickstart mechanism, leading to output like

```
java.net.BindException: Address already in use
```

you already have a running container at IP port 8080. This may be the case if you installed a tomcat server distribution for your operating system, with autostart enabled. Please be sure to stop the application bound to port 8080 before trying to use quickstart.

OK, it's a little more work than that, but not much more. QuickStart assumes the following directory structure:

- webwork
  - lib - all your required libs, usually the ones you would put in WEB-INF/lib
  - webapps
    - mywebapp
      - src
        - java - your java sources that would normally be compiled to WEB-INF/classes
        - webapp
          - WEB-INF
            - classes - any additional configuration if you'd like
            - web.xml
  - webwork.jar
  - launcher.jar

You can quickly get started by copying one of the existing webapps in the WebWork distribution.

Once you have it up and running, you are free to change your classes, JSPs, template files, and other files on the fly - all without compiling or redeploying. Some files, such as web.xml, will require that you restart QuickStart for the changes to take affect. Similarly, some frameworks, such as Hibernate, do not offer the full class-reloading support that WebWork does. Your mileage may vary, but we think no matter what you'll love developing in QuickStart.

Advanced Deployment

Don't have a directory structure like the one laid out? Want to use a port other than 8080? No problem! There are two options for you:

- Apply additional command-line options
- Use a `quickstart.xml` configuration file

Additional Command-line Options

While the quickstart:xxx shorthand is nice, it often doesn't work for many people beyond the initial WebWork distribution packaging. So QuickStart allows you to specify three options from the command line:

- Context
- Webapp directory
- Source directory

Suppose your project layout is the following:

- project
    - lib - all your required libs, usually the ones you would put in WEB-INF/lib
        - src
            - java - your java sources that would normally be compiled to WEB-INF/classes
            - webapp
                - WEB-INF
                    - classes - any additional configuration if you'd like
                    - web.xml

You could launch your application using QuickStart by executing the command:

```
java -jar lib/webwork.jar quickstart /project src/webapp src/java
```

Using the quickstart.xml File

Sometimes the command line options still aren't enough. For whatever reasons, port 8080 might not be enough, or you may need to extend other configurations. Or perhaps your libs are not in your project but instead are in some other directory (very common if you use Maven to build your project). To help out, QuickStart provides a configuration file that let's you tweak how the deployment happens and how it is configured as much as you'd like. Consider the sample quickstart.xml file:

If you use this deployment technique, **you must remember** that quickstart.xml must be in the same working directory in which you execute the `java -jar webwork.jar quickstart` command. You don't need to pass any additional command line arguments to QuickStart, but you must have this file in your working directory.

How It Works

QuickStart works by using the combination of WebWork's "share nothing" (or rather, "share very little") architecture, an embedded Jetty server, some advanced class loading, and the Eclipse Java compiler (don't worry, the Eclipse IDE is not required!)

Running webwork.jar bootstraps the classpath and includes every jar found in the **lib** directory. It also includes webwork.jar, of course. It then invokes the QuickStart application. This, in turn, starts a Jetty server that is configured to the webapp specified in the **quickstart:xxx** argument.

The Jetty server's context ClassLoader is specified as a custom ClassLoader that looks at the source files in **webapps/xxx/src/java** and compiles them on the fly. These classes are also reloaded whenever a change is detected.

Because WebWork creates a new action on every request, reloading the classes works great. You are free to change the entire class schema (methods, fields, inheritance, etc). Because none of the objects are cached or stored in long-term storage, you usually won't run into any problems here.

Running in Your IDE

Running QuickStart from your IDE is no different than running it from the command line. The only difference is that you need to set up the structure and classpath in your IDE properly. It doesn't really matter what is in the **classpath** as long the WebWork jar is included. Pay close attention to your **working directory**.

An example of what IntelliJ IDEA looks like when launching QuickStart from within the WebWork project is included for reference (click for a larger view):

Common Pitfalls

While WebWork is pretty good about making class reloading in QuickStart easy, other libraries and code are not. As a general rule of thumb, if any objects have long term state (singleton, session scope, etc), they will not be reloaded. The reloaded classes will only take affect after a new instance has been created with the new keyword or through reflection.

For example, Hibernate has been found to store references to the objects it persists for long periods of time because of it's caching mechanism. It also happens to hold a reference to the Class instance itself. This makes it very difficult, if not impossible, to allow you to change your models on the fly.

Most problems will manifest themselves through a ClassCastException, or some other weird class-related error. You may even find yourself banging your head against the wall because some Foo instance can't be cast to the Foo class. This is the biggest challenge with using QuickStart and can best be mitigated by using libraries and code that share very little state.

A final word of warning: QuickStart is not meant for production use, or even to be used as the sole environment for application development. Rather, it is meant to help you quickly develop proof-of-concepts and see results quickly. We recommend you always at least test in other applications servers, such as Tomcat, Resin, or even standalone Jetty.

Related Tools

This page last changed on Mar 30, 2006 by scud.

WebWork has several tools that can help when developing WebWork-based applications. Some of these, such as the Config Browser, are used within your web applications. Others, such as those run from in webwork.jar, are used at build-time.

Runtime tools

1. Config Browser

Build-time tools

WebWork comes with various related tools included in the webwork jar file. You can access these tools by simply unpacking the WebWork distribution and running **java -jar webwork.jar**. WebWork will automatically include all jars in the same directory as the webwork.jar file as well as all jars in the lib directory. This means you can invoke these tools either from within the standard directory structure found in the WebWork distribution, or from within your WEB-INF/lib directory.

You can access the help information for these tools by simply running the jar without any arguments.

1. SiteGraph
2. QuickStart

Config Browser

This page last changed on Mar 30, 2006 by scud.

The config browser is a simple tool to help view your WebWork action configuration at runtime. It is very useful when debugging problems that could be related to configuration. To install it, you need to follow these two steps:

1. Add the FreeMarker dependencies to your web application
2. Add the following to your xwork.xml: `<include file="config-browser.xml"/>`

Once you have done this, you can access the tool but going to the action named index in the config-browser namespace. In most cases (if you are using the default ActionMapper), the URL is something like http://localhost:8080/starter/config-browser/index.action.

SiteGraph

This page last changed on Mar 30, 2006 by scud.

Introduction

WebWork comes with a utility called SiteGraph. SiteGraph is used to generate graphical diagrams representing the flow of your web application. It does this by parsing your configuration files, action classes, and view (JSP, Velocity, and FreeMarker) files. An example of a typical output of SiteGraph is provided (for the full size, click [here|^example.gif]):

!example.gif|align=center, width=400!

Additional information can be found in the JavaDocs:

SiteGraph is a tool that renders out GraphViz-generated images depicting your WebWork-powered web application's flow. SiteGraph requires GraphViz be installed and that the "dot" executable be in your command path. You can find GraphViz at http://www.graphviz.org.

Understanding the Output

There are several key things to notice when looking at the output from SiteGraph:

- Boxes: those shaded red indicate an action; those shaded green indicate a view file (JSP, etc).
- Links: arrows colored green imply that no new HTTP request is being made; black arrows indicate a new HTTP request.
- Link labels: labels may sometimes contain additional useful information. For example, a label of **href** means that the link behavior is that of a hyper-text reference. The complete label behaviors are provided:
  - **href** - a view file references an action by name (typically ending with the extension ".action")
  - **action** - a view file makes a call to the Action tag
  - **form** - a view file is linked to an action using the Form tag
  - **redirect** - an action is redirecting to another view or action
  - **! notation** - a link to an action overrides the method to invoke

Requirements

SiteGraph requires that your view files be structured in a very specific way. Because it has to read these files, only certain styles are supported. The requirements are:

- The JSP tags must use the "ww" namespace.
  - In JSP: <ww:xxx/>
  - In FreeMarker: <@ww.xxx/>
  - In Velocity: N/A
- Use of the Form tag and Action tag must be linking directly to the action name (and optional namespace). This means that <ww:form action="foo"/> is OK, but <ww:form action="foo.action"/> is not.
- All code is expected to be using the Alt Syntax.

Setting up

SiteGraph is built in to WebWork, so if you're up and running with WebWork, you don't need to do anything additional java packages. However, SiteGraph does require the "dot" package by GraphViz.

You'll need to download the latest version of GraphViz and make sure that the dot executable (dot.exe in Windows) is in your command path. In Windows the GraphViz installer typically automatically adds dot.exe to your path. However, you may need to do this by hand depending on your system configuration.

Usage

You can use SiteGraph with the following command:

```
java -cp ... -jar webwork.jar
     sitegraph
     -config CONFIG_DIR
     -views VIEWS_DIRS
     -output OUTPUT
     [-ns NAMESPACE]
```

Where:

```
Usage: -config CONFIG_DIR -views VIEWS_DIRS -output OUTPUT [-ns NAMESPACE]
       CONFIG_DIR => a directory containing xwork.xml
       VIEWS_DIRS => comma seperated list of dirs containing JSPs, VMs, etc
       OUPUT      => the directory where the output should go
       NAMESPACE  => the namespace path restriction (/, /foo, etc)
```

You must supply the correct classpath when invoking the SiteGraph tool. Specifically, the XWork, WebWork, and their dependencies must be included in the classpath. Futhermore, **you must also include your action class files referenced in xwork.xml**. Without the proper class path entries, SiteGraph will not function properly.

Once you have run SiteGraph, check the directory specified in the "output" argument (OUTPUT). In there you will find two files: **out.dot** and **out.gif**. You may immediately open up **out.gif** and view the web application flow. However, you may also wish to either run the **out.dot** file through a different GraphVis layout engine (neato, twopi, etc), so the original dot file is provided as well. You may also wish to edit the dot file before rendering the final flow diagram.

Automatic Execution

Some advanced users may wish to execute SiteGraph from within their application - this could be required if you are developing an application that supports WebWork plugin capabilities. This can easily be done. See the JavaDocs for more info:

If you wish to use SiteGraph through its API rather than through the command line, you can do that as well. All you need to do is create a new SiteGraph instance, optionally specify a Writer to output the dot content to, and then call #prepare().

The command line version of SiteGraph does exactly this (except for overriding the Writer):

```
SiteGraph siteGraph = new SiteGraph(configDir, views, output, namespace);
```

```
    siteGraph.prepare();
    siteGraph.render();
```

Result Types

This page last changed on May 01, 2006 by zjumty.

关于Result如何配置参见Result配置

概况

Result类型 是在Action执行完,一个结果返回后决定发生什么事的类。开发者可以自由的根据他们的应用和环境的需要创建自己的Result类型。例如在WebWork2中，Servlet和Velocity结果类型已经被创建用来显示web应用程序的画面。

注意: 所有的webwork内建的Result类型都实现了com.opensymphony.xwork.Result接口. 这个接口是所有action执行结果的通用接口,不管这个结果是用来显示一个网页还是产生一个email,发送一个JMS消息,等.

Result类型配置中定义了一些类,把它们映射为action配置中可以引用的名字. 也就是为这些类创建便于记忆的键-值对.

```
...
<result-types>
    <result-type name="dispatcher"
class="com.opensymphony.webwork.dispatcher.ServletDispatcherResult" default="true"/>
    <result-type name="redirect"
class="com.opensymphony.webwork.dispatcher.ServletRedirectResult"/>
    <result-type name="velocity" class="com.opensymphony.webwork.dispatcher.VelocityResult"/>
    <result-type name="chain" class="com.opensymphony.xwork.ActionChainResult"/>
    <result-type name="xslt" class="com.opensymphony.webwork.views.xslt.XSLTResult"/>
    <result-type name="jasper"
class="com.opensymphony.webwork.views.jasperreports.JasperReportsResult"/>
    <result-type name="freemarker"
class="com.opensymphony.webwork.views.freemarker.FreemarkerResult"/>
    <result-type name="httpheader"
class="com.opensymphony.webwork.dispatcher.HttpHeaderResult"/>
    <result-type name="stream" class="com.opensymphony.webwork.dispatcher.StreamResult"/>
    <result-type name="plaintext" class="com.opensymphony.webwork.dispatcher.PlainTextResult"
/>
</result-types>
...
```

```
<include file="webwork-default.xml"/>

<package name="myPackage" extends="default">
  <action name="bar" class="myPackage.barAction">
    <!-- default result type is "dispatcher" -->
    <!-- default result name is "success" -->
    <result>foo.jsp</result>
    <result name="error">error.jsp</result>
    </result>
  </action>
</package>
```

Result类型

Webwork提供了一些com.opensymphony.xwork.Result接口的实现来使你的action可以容易的用户交互.这些Result类型包括
:

- Chain Result - 用于 Action Chaining
- Dispatcher Result - 用于 JSP 整合
- FreeMarker Result - 用于 FreeMarker 整合

- HttpHeader Result - 用于控制特殊的HTTP行为
- JasperReports Result - 用于 JasperReports 整合
- Redirect Result - 用于直接跳转到例外的URL
- Redirect Action Result - 用于直接跳转到另外的action
- Stream Result - 用于向浏览器返回一个InputStream (一般用于文件下载)
- Velocity Result - 用于 Velocity 整合
- XSL Result - 用于 XML/XSLT 整合
- PlainText Result - 用于显示某个页面的原始的文本 (例如 jsp, html 等)

Result定义在xwork xml配置文件(xwork.xml)中的action标签里。如果location参数是result标签的唯一的参数，你可以这样简化:

```
<action name="bar" class="myPackage.barAction">
  <result name="success" type="dispatcher">
    <param name="location">foo.jsp</param>
  </result>
</action>
```

或者

```
<action name="bar" class="myPackage.barAction">
  <result name="success" type="dispatcher">foo.jsp</result>
</action>
```

如果你扩展了webwork-default.xml, 那么默认的返回类型是"dispatcher". 同样,如果你没有指定result的名字,默认将是"success". 就是说你可以如下简化:

```
<action name="bar" class="myPackage.barAction">
  <result>foo.jsp</result>
</action>
```

注意 : Parse属性允许的location参数作为表达式.例如你可以这样用:

```
<result name="success" type="redirect">/displayCart.action?userId=${userId}</result>
```

注意 : 你也可以指定全局Result以便在多个action中使用. 当要为很多不同的action添加相同的结果是这样会节省时间. Result标签和全局Result的更多信息,参见Result配置部分.

Chain Result

This page last changed on May 01, 2006 by zjumty.

这个result调用另外的一个action，连接自己的拦截器栈和result。

参数

- actionName (默认) - 被调用的action的名字
- namespace - 被调用的action的名称空间. 如果名称空间为空，这默认为当前名称空间
- method - 用于指定目标action的另一个方法被调用. 如果空，默认为excute方法

例子

```
<package name="public" extends="webwork-default">
    <!-- Chain creatAccount to login, using the default parameter -->
    <action name="createAccount" class="...">
        <result type="chain">login</result>
    </action>

    <action name="login" class="...">
        <!-- Chain to another namespace -->
        <result type="chain">
            <param name="actionName">dashboard</param>
            <param name="namespace">/secure</param>
        </result>
    </action>
</package>

<package name="secure" extends="webwork-default" namespace="/secure">
    <action name="dashboard" class="...">
        <result>dashboard.jsp</result>
    </action>
</package>
```

Dispatcher Result

This page last changed on May 03, 2006 by zjumty.

Include或者Forward到一个视图(通常是JSP). 在背后WebWork是用RequestDispatcher来实现的, 在RequestDispatcher里目标servlet/JSP得到与原始的servlet/JSP同样的request/response对象. 因此, 你可以用request.setAttribute()在他们之间传递数据 - WebWork的action可用.

Result可以有三种执行方式:

- 如果我们在一个JSP的范围内(PageContext对象可用), PageContext的include(String)方法会被调用.
- 如果没有PageContext对象, 并且我们也不在任何形式的include中(在request的属性中没有"javax.servlet.include.servlet_path"), 那么调用RequestDispatcher的forward方法
- 否则调用RequestDispatcher的include方法

参数

- location (默认) - 执行后转到的地方(如 jsp).
- parse - 默认为true. 如果设置为false, location 参数就不会被解析为Ognl表达式.

例子

```
<result name="success" type="dispatcher">
  <param name="location">foo.jsp</param>
</result>
```

FreeMarker Result

This page last changed on May 04, 2006 by zjumty.

用Freemarker模板引擎呈现页面视图.另外com.opensymphony.webwork.dispatcher.ServletDispatcherResult的dispatcher结果类型可以和WebWork的FreemarkerServlet一起使用.

由FreemarkerManager类配置模板加载器, 所以模板的位置可以有两种:

- 相对于web根文件夹. 如 /WEB-INF/views/home.ftl
- classpath下的资源. 如 com/company/web/views/home.ftl

参加WebWork Freemarker Support.

参数

- location (默认) - 模板的位置.
- parse - 默认为true. 如果设置为false, location参数不会被当作Ognl表达式解析.
- contentType - 如果不指定默认为"text/html"

例子

```
<result name="success" type="freemarker">foo.ftl</result>
```

This page last changed on May 05, 2006 by zjumty.

WebWork中的Freemarker支持

我们可以通过webwork的result类型freemarker来呈现Freemarker页面. 或者dispacther 结果类型加上WebWork的FreemarkerServlet来呈现.

这篇文档主要讲使用freemarker结果类型, 因为它是被推荐使用的. 下面也有将如何使用FreemarkerServlet

配置你的action使用freemarker结果类型

Freemarker结果类型定义在webwork-default.xml中, 所以正常情况下你只需要include它,然后把你的结果的type属性设置为freemarker

```
<include file="webwork-default.xml"/>
...
<action name="test" class="package.Test">
  <result name="success" type="freemarker">/WEB-INF/views/testView.ftl</result>
</action>
...
```

属性获取

你的action的属性会被自动的获取- 就像velocity页面一样

例如${name}会调用stack.findValue("name"), 其实背后通常又是调用action.getName().

获取变量有一个搜索过程, 顺序搜索以下范围,直到找到需要的值:

- freemarker variables
- value stack
- request attributes
- session attributes
- servlet context attributes

上下文环境中的变量

The following variables exist in the freemarer views

在Freemarker页面中存在如下变量:

- req - 当前的HttpServletRequest对象
- res - 当前的HttpServletResponse对象
- stack - 当前的OnglValueStack对象
- ognl - OgnlTool 实例
    - 这个类由一些有用的方法可以直接执行OGNL表达式, 和一个方法可以生成一个<ww:select>模式选择列表 (例如, 取得list属性的名字, 一个listKey和listValue)
- webwork - FreemarkerWebWorkUtil实例

- action - 当前的WebWork的action对象
- exception - 可选的异常实例, 如果视图是一个JSP异常或者Servlet异常视图

最近的发行版(2.1以后)中的Freemarker配置

要配置webwork用的freemarker引擎, 只需要在classpath下添加一个freemarker.properties文件. 支持的属性都是freemarker的配置对象 - 可以在freemarker的文档中查找到. 这些属性可以同时给freemarker结果类型和webwork的FreemarkerServlet使用.

```
default_encoding=ISO-8859-1
template_update_delay=5
locale=no_NO
```

使用webwork UI标签 - 或者JSP标签库

Freemarker有直接使用任何JSP标签库的能力. 你可以在Freemarker中使用JSP标签库即使
a) 你的Servlet容器不支持JSP, 或
b) 你没有在web.xml中配置标签库- 注意下面的例子中我们如何引用一个标签库定义文件的,这个文件是基于webapp的绝对URL, 所以不需要在web.xml中配置

```
<#assign ww=JspTaglibs["/WEB-INF/webwork.tld"] />

<@ww.form method="'post'" name="'inputform'" action="'save.action'" >
    <@ww.hidden name="'id'" />
    <@ww.textarea label="'Details'" name="'details'" rows=5 cols=40 />
    <@ww.submit value="'Save'" align="center" />
</@ww.form>
```

注意: 标签数值属性必须是数字, 不是字符. 像上面的rows和cols属性, 如果你使用cols="40" 你会得到一个异常. 除此之外freemarker标签的容器和你想象的一样. 译者注: 如果你用ww2.2以后的版本, 就像用JSP标签一样就可以了, 不用像上述的那样.

使用FreemarkerServlet

freemarker.jar中提供的FreemarkerServlet可以被拿出来单独使用, 尽管如此, 它不提供任何webwork独有的功能, 如环境变量, 属性获取等. 因此webwork提供自己的Servlet来支持这一整合.

在web.xml中注册FreemarkerServlet

要使用freemarker作为表现层技术, 必须配置webwork2的FreemarkerServlet, 映射你的模版用的文件的扩展名.

```
<servlet>
  <servlet-name>freemarker</servlet-name>
  <servlet-class>com.opensymphony.webwork.views.freemarker.FreemarkerServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>freemarker</servlet-name>
  <url-pattern>*.ftl</url-pattern>
</servlet-mapping>
```

配置Actions来用这个Servlet(xwork.xml配置)

要使用Freemarker页面，只需要用dispatcher结果类型，指定location为模版文件位置

```
<action name="test" class="package.Test">
  <result name="success" type="dispatcher">/WEB-INF/views/testView.ftl</result>
</action>
```

扩展这个Servlet

注意：这个文档需要修订，因为在完成时FreemarkerServlet已经改变了.

Freemarker Servlet的详细信息可以到freemarker站点上查看

小心当扩展com.opensymphony.webwork.views.freemarker.FreemarkerServlet，重写

```
protected TemplateModel createModel(
    ObjectWrapper wrapper,
    ServletContext servletContext,
    HttpServletRequest request,
    HttpServletResponse response)
```

调用super.createModel(...)方法，然后用新模型包装以解决action的属性获取问题

HttpHeader Result

This page last changed on May 04, 2006 by zjumty.

根据ValueStack返回自定义的HTTP header

参数

- status - HTTP响应的状态码.
- parse - 默认为true. 如果设置为false, header参数不会被当作Ognl表达式解析.
- headers - header中的值.

例子

```
<result name="success" type="httpheader">
  <param name="status">204</param>
  <param name="headers.a">a custom header value</param>
  <param name="headers.b">another custom header value</param>
</result>
```

JasperReports Result

This page last changed on May 04, 2006 by zjumty.

根据指定的格式生成JasperReports报表,如果没有指定格式默认为PDF格式

参数

- location (默认) - 编译过的jasper报表定义文件(foo.jasper)的位置, 相对于当前URL.
- dataSource (必需) - 从ValueStack中取数据源的Ognl表达式 (一般情况下数据源为List).
- parse - 默认为true. 如果设置为false, location参数不会被当作Ognl表达式解析.
- format - 报表生成格式. 有效的类型见JasperReportConstants. 如果不指定格式, 默认使用PDF.
- contentDisposition - 布局(默认为"inline", 值一般为filename="document.pdf").
- documentName - 文档的名字(会生成http头 Content-disposition = X; filename=X.[format]).
- delimiter - 生成CSV报表的分隔符. 默认为字符",".
- imageServletUrl - 当以context page为前缀时可以返回报表图片的url的名字.

这个Result拥有WebWorkResultSupport的所有规则. 尤其是当"parse"参数没有被设置为false时,所有的参数可以被解析为Ognl表达式.

例子

```
<result name="success" type="jasper">
  <param name="location">foo.jasper</param>
  <param name="dataSource">mySource</param>
  <param name="format">CSV</param>
</result>
```

或者pdf:

```
<result name="success" type="jasper">
  <param name="location">foo.jasper</param>
  <param name="dataSource">mySource</param>
</result>
```

PlainText Result

This page last changed on May 05, 2006 by zjumty.

这个Result可以发送普通文本内容. 当你要显示一个JSP或者Html文件的原始内容时这个Result会很有用.

参数

- location (默认) = 要显示普通文本的文件(jsp/html)的位置.
- charSet (可选) = 使用的字符集. 这个字符集用来设置被阅读器显示时的响应类型(如Content-Type=text/plain; charset=UTF-8). 字符集的例子UTF-8, ISO-8859-1 etc. Content-Type=text/plain; charset=UTF-8

例子

```
<action name="displayJspRawContent" >
  <result type="plaintext">/myJspFile.jsp</result>
</action>


<action name="displayJspRawContent" >
  <result type="plaintext">
     <param name="location">/myJspFile.jsp</param>
     <param name="charSet">UTF-8</param>
  </result>
</action>
```

Redirect Action Result

This page last changed on May 04, 2006 by zjumty.

这个Result使用ActionMapperFactory提供的ActionMapper来重定位浏览器的URL来调用指定的action和(可选的)namespace. 这个Result比ServletRedirectResult要好.因为你不需要把URL编码成xwork.xml中配置的ActionMapper提供的模式. 这就是说你可以在任意点上改变URL模式而不会影响你的应用程序. 因此强烈推荐使用这个Result而不是标准的redirect result来解决重定位到某个action的情况.

详细信息参见 ActionMapper

参数

- actionName (默认) - 重定位到的action名
- namespace - action的名称空间. 如果为null，则为当前名称空间

例子

```
<package name="public" extends="webwork-default">
    <action name="login" class="...">
        <!-- Redirect to another namespace -->
        <result type="redirect-action">
            <param name="actionName">dashboard</param>
            <param name="namespace">/secure</param>
        </result>
    </action>
</package>

<package name="secure" extends="webwork-default" namespace="/secure">
    <-- Redirect to an action in the same namespace -->
    <action name="dashboard" class="...">
        <result>dashboard.jsp</result>
        <result name="error" type="redirect-action">error</result>
    </action>

    <action name="error" class="...">
        <result>error.jsp</result>
    </action>
</package>
```

Redirect Result

This page last changed on May 04, 2006 by zjumty.

调用{@link HttpServletResponse#sendRedirect(String) sendRedirect}方法来转到指定的位置. HTTP响应被告知使浏览器直接跳转到指定的位置(产生客户端的一个新请求). 这样做的结果会使刚刚执行的action(包括action实例,action中的错误消息等)丢失, 不再可用. 这是因为action是建立在单线程模型基础上的. 传递数据的唯一方式就是通过Session或者可以为Ognl表达式的web参数(url?name=value)

Parameters

- location (默认) - action执行后跳转的地址.
- parse - 默认为true. 如果设置为false, location参数不会被当作Ognl表达式解析.

例子

```
<result name="success" type="redirect">
  <param name="location">foo.jsp</param>
  <param name="parse">false</param>
</result>
```

Stream Result

This page last changed on May 05, 2006 by zjumty.

直接向HttpServletResponse发送原始数据(通过InputStream)的自定义Result类型. 它在让用户下载东西时很有用

参数

- contentType - 发送到web浏览器的流的mime-type (默认为text/plain).
- contentLength - 以Byte计算的流的长度(浏览器显示进度条).
- contentDispostion - 内容部属头的值, 指定文件名(默认为inline, 值一般是 filename="document.pdf".
- inputName - action链中的InputStream属性的名字 (默认为inputStream).
- bufferSize - 从输入复制到输出的缓冲区的大小 (默认为1024).

例子

```
<result name="success" type="stream">
  <param name="contentType">image/jpeg</param>
  <param name="inputName">imageStream</param>
  <param name="contentDisposition">filename="document.pdf"</param>
  <param name="bufferSize">1024</param>
</result>
```

Velocity Result

This page last changed on May 05, 2006 by zjumty.

这个Result用Servlet容器中的JspFactory模拟JSP执行环境来显示Velocity模板，然后以流的形式输出到Servlet Output.

参数

- location（默认）- 模板的位置.
- parse - 默认为true. 如果设置为false, location参数不会被当作Ognl表达式解析.

这个Result拥有WebWorkResultSupport所有的规则

例子

```
<result name="success" type="velocity">
  <param name="location">foo.vm</param>
</result>
```

XSL Result

This page last changed on May 05, 2006 by zjumty.

XLSTResult 用XSLT来转换action对象到XML. 最近版本已经修改过Xalan处理的缺陷. 当用Xalan你可能会注意到,你需要有非常简单的StyleSheet,像这样:

```
<xsl:template match="/result">
   <result />
 </xsl:template>
```

然后Xalan仍然会遍历你的action的每个属性.
如果你有双链接的对象, 即使你的样式表并不是创建来处理所有内容的,Xalan也会分析整个对象树. 因为当前的Xalan是在处理前先把每个东西都转换成内部的DTM模型.

这就是为什么会有个循环消除器在处理时为每个对象属性创建索引.如果它发现一些对象的属性已经访问过了, 他就不会再访问了.也就是说,你有两个对象x和y,有下列属性集(伪代码):

```
x.y = y;
 and
 y.x = x;
 action.x=x;
```

由于修改,基于x的结果XML文档应该是:

```
<result>
   <x>
     <y/>
   </x>
 </result>
```

如果没有上述机制，结果就是无止境的x/y/x/y/x/y/...元素.

XSLTResult编码也尝试处理这样的事实: DTM模型创建的方式是子节点先于兄弟节点被处理。结果是如果有一个对象x被同时设置为action的x属性和一个非常深的action属性a. 那么仅会出现在a下面而不是x下面. 这不是我们所期望的, 这也是为什么XSLTResult允许对象在一些地方重复出现.

有时这种对象网络非常密集, 你可能发现你的简单的样式表的执行消耗了大量时间. 为了帮助你处理Xalan的这个缺陷,你可以为元素路径(xpath)上附加regexp过滤器.

注意: 你的.xsl文件的根匹配必须名为result.
这个例子将会根据action类中的getUsername输出用户名:

```
<xsl:template match="result">
   <html>
   <body>
   Hello <xsl:value-of select="username"/> how are you?
   </body>
   <html>
 <xsl:template/>
```

在下面的例子中,XLST结果仅仅会访问到action的属性而不访问他们的子属性. 它会跳过名字中有"hugeCollection"的所有属性. 元素的路径首先比较excludingPattern, 如果符合这不再处理. 然后会比较matchingPattern如果符合再处理.

```
<result name="success" type="xslt">
  <param name="location">foo.xslt</param>
  <param name="matchingPattern">^/result/[^/*]$</param>
  <param name="excludingPattern">.*(hugeCollection).*</param>
</result>
```

参数

- location (默认) - xslt文件的位置
- parse - 默认为true. 如果设置为false, location参数不会被当作Ognl表达式解析.
- matchingPattern - 想要的元素的模式, 默认接受所有.
- excludingPattern - 不想要的元素的模式, 默认不拒绝任何.

webwork.properties 相关配置:

webwork.xslt.nocache - 默认为否. 如果设置为true禁止样式表缓存. 适于开发,不适于生产环境.

例子

```
<result name="success" type="xslt">
  <param name="location">foo.xslt</param>
  <param name="matchingPattern">^/result/[^/*]$</param>
  <param name="excludingPattern">.*(hugeCollection).*</param>
</result>
```

Struts Action 2.0 Information

This page last changed on Apr 04, 2006 by darren.

在WebWork 2.2.2发布之后，WebWork和Struts的社区将会开始合并Webwork和Struts，最终会成为 Struts Action 2.0,你可以通过访问项目首页找到更多关于这个合并的信息以及最近的状态：

http://incubator.apache.org/projects/webwork2.html

Style Guide

This page last changed on May 19, 2006 by scud.

This page contains my suggestions on documentation style. I will try to demonstrate my suggestions by writing a document that justifies them. I'm an advocate of **learning by example**. As Mark Twain said, "Few things are harder to put up with than the annoyance of a good example" (Samuel Langhornne Clemens (1835-1910)).

Pulling content from CVS

A large part of the WebWork 2.2 documentation effort is to make documentation easier to handle in the future. At on the onset of this effort, everyone agreed that pulling as much source code, examples, and documentation from source control would help greatly. As such, we've installed a modified version of the **snippet macro** in the OpenSymphony wiki.

Important!
Whenever you write documentation, ask yourself if you can somehow have this documentation checked in to source control in the form of example code or JavaDocs. This will make the documentation much more likely to be useful for years to come.

The standard way to use it is to do the following:

{snippet:id=description|javadoc=true|url=com.opensymphony.xwork.interceptor.LoggingInterceptor}

or

{snippet:id=example|javadoc=true|lang=xml|url=com.opensymphony.xwork.interceptor.LoggingInterceptor}

or

{snippet:id=sitegraph-usage|lang=none|url=webwork/src/java/com/opensymphony/webwork/sitegraph/sitegraph-usage.txt}

or

{snippet:id=ajax-validation-example|lang=xml|url=webwork/webapps/ajax/src/webapp/lesson1/example.jsp}

Where:

- id is the name of the snippet (*required).
- url is the URL where the snippet can be found (**required**).
- lang is the language that the code block should be required as. If this snippet is simply text, don't include this parameter and the content will be printed outside of a code block.
- javadoc indicates if the content is within a JavaDoc block. If this is set to true, then the preceeding "* " (asterisk-space) characters will be stripped before printing the content out. Also, the content is assumed to the HTML escaped already and therefore won't be escaped again.

All snippets are marked off by the pattern **START SNIPPET**: XXX and **END SNIPPET**: XXX where XXX is the name of the snippet that is assigned in the id attribute of the macro. The URL is typically a location that points to the project's source control contents.

A note about URLs

As you probably noticed in the examples, there are several formats that the URL patterns can be in. A fully-qualified URL is always allowed, though that is often not practical. We've customized the macro to be a bit more intelligent with the URL attribute:

- If the URL appears to be a class, we assume it lives in src/java, convert all the dots to slashes, and then append .java to it.
- If the URL doesn't start with "http", then it is assumed to start with https://opensymphony.dev.java.net/source/browse/*checkout*/ , as you saw in the third example.

Note that the short-hand class notation will only work for top-level projects (WebWork, OSWorkflow, etc) and not any sub-projects within the projects, such as example webapps in WebWork. If you wish to include content from a class in a sub-project, you'll need to list out the full path, like in the fourth example.

A note about snippet markers

All snippet markers should be commented out if possible. How they are commented out depends on where the snippet is. If the snippet is for HTML or XML, you can do:

```
<!-- START SNIPPET: xxx -->
...
<!-- END SNIPPET: xxx -->
```

If the snippet is in Java code, you can do:

```
if (true != false) {
    // START SNIPPET: xxx
    System.out.println("This is some silly code!");
    // END SNIPPET: xxx
}
```

If the snippet is found in JavaDocs, you should use HTML comments as they won't render in the JavaDocs. For XML examples in JavaDocs (see Timer Interceptor for an example), it can be a bit tricky. This is because in your JavaDocs you want to use the <pre> tag, but you don't want the wiki to display it. A good technique is to embed the snippet markers inside the <pre> tag:

```
* <pre>
 * <!-- START SNIPPET: example -->
 * &lt;!-- records only the action's execution time --&gt;
 * &lt;action name="someAction" class="com.examples.SomeAction"&gt;
 *     &lt;interceptor-ref name="completeStack"/&gt;
 *     &lt;interceptor-ref name="timer"/&gt;
 *     &lt;result name="success"&gt;good_result.ftl&lt;/result&gt;
 * &lt;/action&gt;
 * <!-- END SNIPPET: example -->
```

About Headings

This section refers to: Notation Guide >> Headings.

Headings should definetly be used. This sections tries to justify why.

First rule: don't use "h1" at the top of each page. The page title serves as the "top level header" already, so there is no need to duplicate that information again. Also, when the docs end up the website, SiteMesh will place a top level "h1" element using the page title.

Document sections

Headings can help you divide your document in sections, subsections, sub-subsections and so forth.

Advantages

Your document becomes more organized.

Disadvantages

If you exaggerate you could fragment your text too much.

Warning

This is definetly an example of this, since this whole "Headings" section has such few paragraphs that it really should have been written in just one section.

Aren't warning boxes neat?

Headings capitalization

I think headers `h1` and `h2` should have all words capitalized (such as "Vitor's Suggestions on Documentation Style" and "About Headings"), but `h3` and smaller would have just the first word (such as "Headings capitalization"). Except, of course, for words that are always capitalized (eg. "Understanding WebWork's importance to OpenSymphony and its community"). This gives even more importance to bigger headings.

Avoid skipping headers

I mean, avoid going from a `h1` directly to a `h3` without using `h2` before. This would be like havin a section 1.1.1 directly below section 1, without the existance of section 1.1.

Handy Hint

One thing that I like to do is leave **five** blank lines before `h1` headings, **three** before `h2`'s and **two** before `h3`'s. Also, in Portuguese (I'm Brazilian), we write small numbers using their names instead of numeric representation (**five** instead of **5**). I don't know if this is also a good practice in written english.

Aren't tip boxes neat?

Be Careful

If you find yourself writting too many `h1` headings in a single page, consider breaking the page into child pages and linking to them.

Aren't note boxes neat?

More on Text Effects

This section refers to: Notation Guide >> Text Effects.

Text effects should be largely used, although I have some questions on some of them. **Strong**, emphasis, and inserted can be used to denote importante parts of a sentence. But I really think inserted should have been called underline in the notation guide. I don't see the point of using ~~deleted~~, since when someone changes a page and deletes stuff, Confluence keeps the old versions in history.

I can't think of a situation in WebWork's doc for superscript and subscript, but it doesn't hurt to mention them. I can't say anything about %span% because I frankly don't know what it does. `Monospaced` is heavily used, for instance, to refer to `webwork-default.xml` file or items in source code examples: `<xmltag />`, `JavaClass` or `javaVariable`.

Boxes vs. Block Quotes

I think boxes and block quotes do the same job, but boxes are better. Therefore, I suggest we don't use block quotes.

Aren't info boxes neat? Aren't them all neat? By now you may have realized I think we should definetly use them...

Colors should be used in very specific cases, or else each documentation writers would color his/her pages the way he/she thinks it's better, and it would look like a mess. One such specific case in which colors can help is when you want them to work as tags or captions. For (a lame) example, in this paragraph, guidelines are in red and justifications are in blue. Yes, it's a really really lame example, I know.

Text Breaks

This section refers to: Notation Guide >> Text Breaks.

Text breaks shouldn't be used. If you'd like paragraphs or headings to have more spacing (before or after), the style sheet should be changed, not the contents. Patrick explained this a long time ago. Other stuff in this section (paragraphs, horizontal ruler, — symbol and – symbol) can be used when necessary.

Links

This section refers to: Notation Guide >> Links.

All types of links can and should be used. I already used a few in this document. Just watch out for links to non-existing pages when writing on the official documentation.

Lists

This section refers to: Notation Guide >> Lists.

Lists can be used for many purposes. Every time we list some things that are in order, ordered lists are used. If they don't have a specific order, unordered lists are the case. List should be nested if needed for a better organization. Unordered lists should be created only with the * (star) notation only, so all pages use the same style of bullet.

- This is an unordered list in star notation;
- Items can have sub-items
    - That can have sub-items
        - That can have sub-items...
            - What is the limit?
- Mixing ordered and unordered lists is possible:
    1. One;
    2. Two;
    3. Three.

List indentation

Use tabs to indent nested lists. This way your page's markup is more readable and easier to maintain.

Images and Icons

This section refers to: Notation Guide >> Images and Notation Guide >> Misc.

External images should be used only when strictly necessary (meaning, don't use images as list bullets or box icons). Also, try to use only images that are very unlikely to be removed from its current URL, to reduce document maintenance. Pay attention on copyright issues too! Attached images are less prone to become missing links, however, we should not clutter the documentation with unnecessary attachments and copyrights are also a issue here.

Example: Cannot resolve external resource into attachment.

Icons are cool in a number of situations. Some of them, such as , , or can make the documentation look professional, but some others, such as and may give a feeling of amateurship and I wouldn't advise them for pages that are exported to form the official documentation.

Tables

This section refers to: Notation Guide >> Tables.

Tables are very useful when lists just don't do it. Meaning: don't write a table when a list suffices. Tables are more organized, because you can align the text in columns. Since the markup text for tables in confluence is not very easy to read, remember complex and big tables are very hard to maintain.

The table below was copied from a reference page on WebWork's configuration (just the first two lines were copied). This is an example where tables are good: a list wouldn't be as organized as this table to display these files and their properties.

File	Optional	Location (relative to webapp)	Purpose
web.xml	no	/WEB-INF/	Web deployment descriptor to include

			all necessary WebWork components
xwork.xml	no	/WEB-INF/classes/	Main configuration, contains result/view types, action mappings, interceptors, etc

Advanced Formatting

This section refers to: Notation Guide >> Advanced Formatting.

I've already made my point about info, warning, tip and note boxes. Other interesting markups are `noformat` and `code`. The former can be used for general purpose text while the latter is used to display example source code, be it HTML, XML, Java or anything that is part of a software solution. When displaying something that has a name, use a title, as the example below demonstrates.

```
/** Hello World class. */
public class HelloWorld {
  /** Main method. */
  public static void main(String[] args) {
    System.out.println("Hello, World!");
  }
}
```

A typical example of `noformat` would be the command line statements to compile and run the code above. We should also standardize terminal notation ({$} for command prompt).

```
$ javac HelloWorld.java

$ java HelloWorld
Hello, World!
```

Do not use tabs inside `noformat` and `code`, use two spaces instead. This way your code is indented but keeps lines short. Large lines should be splitted as to fit in a 800x600 resolution screen without horizontal scroll bars.

Your Comments Please

Please contribute to this page. Let me know if you have a different opinion on something (please justify it). Please warn me if I wrote something wrong or if this proposed Style Guide is missing something. Feel free to correct my english, since I'm not a native speaker.

Tags

This page last changed on Apr 05, 2006 by kaktos.

webwork提供了一套不依赖于显示层技术的标签库。这一章我们将概括性的描述每一个标签， 比如此标签支持的属性，标签的行为等等。 大多数的标签都可以用于所有的模板语言（参考 JSP Tags, Velocity Tags，和 FreeMarker Tags），但是有一些只能用于特定的模板语言。 无论什么时候一个标签不能完全的支持每一个模板，该标签会在参考文档中注明。

webwork中的标签分为两种类型： 通用标签和HTML标签。 除了功能和职责外这两种类型标签最大的区别是HTML标签支持模板（templates）和主题 （themes）。除了基本的参考外，我们将提供个个标签在所支持模板语言中的例子。

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

通用标签(General Tags)

通用标签用于在你的页面被渲染的时候控制流程。它们也可以用于从Action和Value Stack之外提取数据，比如 Internationalization, JavaBeans，以及包含额外的url或者action执行结果。

1. Control Tags 控制标签，负责流程控制，例如 if, else, and iterator.
2. Data Tags 数据标签，数据创建和操作，例如 bean, push, and i18n.

HTML标签

同通用标签不同, HTML标签不过多提供控制结构或逻辑。而是着重于如何使用你 action/value stack或者Data Tags中的数据，并且在html的中呈现出来。所有的HTML标签都有受templates和theme驱动的唯一行为。 如果说普通标签只是简单的作些输出结果 （如果有内容）的话, HTML标签的输出则是因模板（template）而异, 他们常常组合在一个作为一个主题（theme），做实际的渲染输出html的工作。

独特的模板（template）支持功能可以帮助你用HTML标签构建一套丰富的可重用的并且符合你需求的UI组件。 请参考 Themes and Templates 指南了解更多此强大特性的信息。

1. Themes and Templates：解释模板和主题在渲染html标签时如何工作(必读)。
2. Form Tags 提供和表单相关的html输出,例如 form, textfield, 和 select.
3. Non Form Tags 提供和表单无关的html输出, 例如 a, div, 和 tabbedPanel.

特定模板语言的标签支持

WebWork竭力支持你所偏爱的技术，这也是WebWork没有绑定于一种模板语言的原因。 webwork支持几乎所有应用广泛的模板语言甚至还为新语言提供了接口。默认情况下，几乎每一个标签都支持 JSP, Velocity, and FreeMarker。在这些章,你会发现一些范例或者技术教你如何在你选用的模板语言中使用特定的标签。

从WebWork 2.2开始, FreeMarker已经成为webwork team推荐的"标准"模板语言. 这个选择有很多原因, 在不同的论坛中你可以找到, 但有一点相当重要: FreeMarker提供比Velocity更丰富的特性并且提供更好更准确的错误报告机制. JSP也可以用,但是对于一些需要更模块化的应用来说比较困难, 比如在运行期间改变模板或者做上传打包好的webwork的action和template文件模块。

1. JSP Tags

2. Velocity Tags
3. FreeMarker Tags

Control Tags

This page last changed on Apr 05, 2006 by kaktos.

控制标签提供操纵集合以及有条件的生成内容的功能.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

1. if
2. elseIf / elseif
3. else
4. append
5. generator
6. iterator
7. merge
8. sort
9. subset

@see src/META-INF/taglib.tld

append

This page last changed on May 10, 2006 by scud.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

常和Iterator标签一起使用, 功能就是将不同的迭代器组合在一起,使一个迭代器迭代完成后转移到下一个迭代器中继续迭代.

例如有三个迭代器组合在一起 (每一个迭代器有三个元素), 下面说明了新生成的迭代器中的元素是如何被迭代的;

1. 第一个迭代器中的第一个元素
2. 第一个迭代器中的第二个元素
3. 第一个迭代器中的第三个元素
4. 第二个迭代器中的第一个元素
5. 第二个迭代器中的第二个元素
6. 第二个迭代器中的第三个元素
7. 第三个迭代器中的第一个元素
8. 第三个迭代器中的第二个元素
9. 第三个迭代器中的第三个元素

参数

名称	必填	缺省	类型	描述
id	false		Object/String	标明新产生的迭代器在stack context中的名字

例子

```
public class AppendIteratorTagAction extends ActionSupport {

        private List myList1;
        private List myList2;
        private List myList3;


        public String execute() throws Exception {

                myList1 = new ArrayList();
                myList1.add("1");
                myList1.add("2");
                myList1.add("3");

                myList2 = new ArrayList();
                myList2.add("a");
                myList2.add("b");
                myList2.add("c");

                myList3 = new ArrayList();
                myList3.add("A");
                myList3.add("B");
                myList3.add("C");
                return "done";
        }

        public List getMyList1() { return myList1; }
        public List getMyList2() { return myList2; }
        public List getMyList3() { return myList3; }
}
```

```
<ww:append id="myAppendIterator">
        <ww:param value="%{myList1}" />
        <ww:param value="%{myList2}" />
        <ww:param value="%{myList3}" />
</ww:append>
<ww:iterator value="%{#myAppendIterator}">
        <ww:property />
</ww:iterator>
```

else

This page last changed on May 10, 2006 by scud.

⚠ 请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

基本的流程控制.'If'标签可单独使用也可以和'Else If'标签和(或)一个多个'Else'一起使用.

参数

名称	必填	缺省	类型	描述
id	false		Object/String	所指元素的Id. 对于UI和form标签此Id就作为所对应的html标签的id属性

例子

```
<ww:if test="%{false}">
    <div>Will Not Be Executed</div>
</ww:if>
<ww:elseif test="%{true}">
    <div>Will Be Executed</div>
</ww:elseif>
<ww:else>
    <div>Will Not Be Executed</div>
</ww:else>
```

elseIf

This page last changed on May 10, 2006 by scud.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

基本的流程控制.'If'标签可单独使用也可以和'Else If'标签和(或)一个多个'Else'一起使用.

参数

名称	必填	缺省	类型	描述
test	**true**		Boolean	决定If标签内容是否显示的表达式
id	false		Object/String	所指元素的Id. 对于UI和form标签此Id就作为所对应的html标签的id属性

例子

```
<ww:if test="%{false}">
    <div>Will Not Be Executed</div>
 </ww:if>
        <ww:elseif test="%{true}">
    <div>Will Be Executed</div>
 </ww:elseif>
 <ww:else>
    <div>Will Not Be Executed</div>
 </ww:else>
```

generator

This page last changed on May 10, 2006 by scud.

描述

NOTE: JSP-TAG
由val属性提供的值产生一个迭代器.

注意: 所产生的迭代器 **总是** 在value stack的顶端并在此标签结束后被pop出value statck

参数

名称	必填	缺省	类型	描述
count	false		Integer	所生成迭代器中元素的数量
separator	true		String	val属性中的分隔符
val	true		String/Object	用于生成迭代器
converter	false		com.opensymphony.webwork.util.IteratorGenerator.Convert	转换val属性提供的值转换为对象的转换器
id	false		String	如果提供,将代表page context中生成的迭代器

例子

示例1:

生成一个简单的迭代器

```
<ww:generator val="%{'aaa,bbb,ccc,ddd,eee'}">
<ww:iterator>
        <ww:property /><br/>
</ww:iterator>
</ww:generator>
```

生成一个迭代器并通过iterator标签显示结果

示例2:

count属性用来控制产生迭代器中元素数量

```
<ww:generator val="%{'aaa,bbb,ccc,ddd,eee'}" count="3">
<ww:iterator>
```

```
        <ww:property /><br/>
</ww:iterator>
</ww:generator>
```

这会生成一个迭代器,但是在生成的迭代器中只有三个元素可以使用,分别命名为aaa,bbb和ccc,这是因为count属性被设置为3.

示例3:

增加id属性

```
<ww:generator val="%{'aaa,bbb,ccc,ddd,eee'}" count="4" separator="," id="myAtt" />
<%
        Iterator i = (Iterator) pageContext.getAttribute("myAtt");
        while(i.hasNext()) {
                String s = (String) i.next(); %>
                <%=s%> <br/>
<%   }
%>
```

id属性将作为迭代器的key值保存在pageContext中

示例4:

标明converter属性

```
<ww:generator val="%{'aaa,bbb,ccc,ddd,eee'}" converter="%{myConverter}">
<ww:iterator>
                <ww:property /><br/>
        </ww:iterator>
</ww:generator>
```

```
public class GeneratorTagAction extends ActionSupport {

  ....

 public Converter getMyConverter() {
        return new Converter() {
                public Object convert(String value) throws Exception {
                        return "converter-"+value;
                }
        };
 }

  ...

}
```

这个简单的转换器所做的工作就是在每个元素前面加上前缀 ″converter-″ .

if

This page last changed on May 10, 2006 by scud.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

基本的流程控制.'If'标签可单独使用也可以和'Else If'标签和(或)一个多个'Else'一起使用.

参数

名称	必填	缺省	类型	描述
test	**true**		Boolean	决定If标签内容是否显示的表达式
id	false		Object/String	所指元素的Id. 对于UI和form标签此Id就作为所对应的html标签的id属性

例子

```
<ww:if test="%{false}">
    <div>Will Not Be Executed</div>
 </ww:if>
        <ww:elseif test="%{true}">
    <div>Will Be Executed</div>
 </ww:elseif>
 <ww:else>
    <div>Will Not Be Executed</div>
 </ww:else>
```

iterator

This page last changed on May 10, 2006 by scud.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

可以对java.util.Collection,java.util.Iterator类型的值进行迭代

参数

名称	必填	缺省	类型	描述
status	false	false	Boolean	如果提供该属性 每次迭代时候将生成一个IteratorStatus实例并放入堆栈中
value	false		Object/String	迭代源, 或者一个对象被放入最新创建的List
id	false		Object/String	element的id属性

例子

下面的例子通过当前value stack中对象的getDays()返回的值进行迭代并用 <ww:property/> 打印结果.

```
<ww:iterator value="days">
  <p>day is: <ww:property/></p>
</ww:iterator>
```

下面的例子通过Bean标签把值'it'放进了ActionContext.然后iterator标签去取出该值并调用它的getDays()方法. status属性用来创建一个IteratorStatus对象,例子中通过调用该对象的odd方法来调整行的颜色.

```
<ww:bean name="com.opensymphony.webwork.example.IteratorExample" id="it">
  <ww:param name="day" value="'foo'"/>
  <ww:param name="day" value="'bar'"/>
</ww:bean>
<p/>
<table border="0" cellspacing="0" cellpadding="1">
<tr>
  <th>Days of the week</th>
</tr>
<p/>
<ww:iterator value="#it.days" status="rowstatus">
  <tr>
    <ww:if test="#rowstatus.odd == true">
      <td style="background: grey"><ww:property/></td>
    </ww:if>
    <ww:else>
      <td><ww:property/></td>
    </ww:else>
  </tr>
```

```
</ww:iterator>
</table>
```

下面的例子更进一步描述status属性的用途, 通过OGNL从action类中拿到DAO, 对groups和其users进行迭代 (安全的上下文环境). last() 方法标明该对象是否是最后一个,如果不是,需要用逗号分开user.

```
<webwork:iterator value="groupDao.groups" status="groupStatus">
                <tr class="<webwork:if test="#groupStatus.odd == true
">odd</webwork:if><webwork:else>even</webwork:else>">
                        <td><webwork:property value="name" /></td>
                        <td><webwork:property value="description" /></td>
                        <td>
                                <webwork:iterator value="users" status="userStatus">
                                        <webwork:property value="fullName" /><webwork:if
test="!#userStatus.last">,</webwork:if>
                                </webwork:iterator>
                        </td>
                </tr>
</webwork:iterator>
```

下一个例子通过 迭代一个action的collection并把它的所有值传给另一个action. 这里的小技巧是使用操作符 '[0]' . 在这里'[0]'和<ww:property />是一样的效果(但后者显然不能用于param 标签内部).

```
<ww:action name="entries" id="entries"/>
<ww:iterator value="#entries.entries" >
    <ww:property value="name" />
    <ww:property />
    <ww:push value="...">
        <ww:action name="edit" id="edit" >
            <ww:param name="entry" value="[0]" />
        </ww:action>
    </push>
</ww:iterator>
```

merge

This page last changed on May 10, 2006 by scud.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

MergeIteratorTag的组件,作用是合并迭代器,合并后的迭代器迭代时依次调用每一个被合并的迭代器(除非这个迭代器已经迭代完毕)

背后的工作是交给 MergeIteratorFilter完成的

例如有三个list被合并,每一个有三个元素,以下就是调用顺序:

1. 第一个list的第一个元素
2. 第二个list的第一个元素
3. 第三个list的第一个元素
4. 第一个list的第二个元素
5. 第二个list的第二个元素
6. 第三个list的第二个元素
7. 第一个list的第三个元素
8. 第二个list的第三个元素
9. 第三个list的第三个元素

参数

名称	必填	缺省	类型	描述
id	false		Object/String	标明新产生的迭代器在stack context中的名字

例子

```
public class MergeIteratorTagAction extends ActionSupport {

private List myList1;
private List myList2;
private List myList3;

public List getMyList1() {
        return myList1;
}

public List getMyList2() {
        return myList2;
}

public List getMyList3() {
        return myList3;
}


public String execute() throws Exception {
        myList1 = new ArrayList();
        myList1.add("1");
        myList1.add("2");
        myList1.add("3");

        myList2 = new ArrayList();
        myList2.add("a");
        myList2.add("b");
        myList2.add("c");

        myList3 = new ArrayList();
        myList3.add("A");
        myList3.add("B");
        myList3.add("C");

        return "done";
}
}
```

```
<ww:merge id="myMergedIterator1">
        <ww:param value="%{myList1}" />
        <ww:param value="%{myList2}" />
        <ww:param value="%{myList3}" />
</ww:merge>
<ww:iterator value="%{#myMergedIterator1}">
        <ww:property />
</ww:iterator>
```

sort

This page last changed on May 10, 2006 by scud.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

NOTE:JSP-TAG
用传入的 Comparator对List进行排序.如果提供id属性,该属性作为排序后的list在PageContext中的标识. 排序后的list放在堆栈顶端,并在此tag结束后弹出.

参数

名称	必填	缺省	类型	描述
comparator	true		java.util.Comparator	用来做比较的comparator
source	false		Object/String	迭代的内容来源
id	false		String	tag元素的id属性

例子

用法 1:

```
<ww:sort comparator="myComparator" source="myList">
      <ww:iterator>
                <!-- do something with each sorted elements -->
                <ww:property value="..." />
      </ww:iterator>
 </ww:sort>
```

用法 2:

```
<ww:sort id="mySortedList" comparator="myComparator" source="myList" />

 <%
    Iterator sortedIterator = (Iterator) pageContext.getAttribute("mySortedList");
    for (Iterator i = sortedIterator; i.hasNext(); ) {
        // do something with each of the sorted elements
    }
 %>
```

subset

This page last changed on May 10, 2006 by scud.

描述

NOTE: JSP-TAG
取一个迭代器的子集.内部实现是通过com.opensymphony.webwork.util.SubsetIteratorFilter.

参数

名称	必填	缺省	类型	描述
count	false		Integer	子集中的元素个数
source	false		Object/String	源集合
start	false		Integer	子集在源集合中的开始索引(如从0开始)
decider	false		com.opensymphony.webwork.util.SubsetIteratorFilter.Deci	用于判断是否某特定元素包含在子集中
id	false		String	标签元素的id属性

例子

```
public class MySubsetTagAction extends ActionSupport {
      public String execute() throws Exception {
                   l = new ArrayList();
                   l.add(new Integer(1));
                   l.add(new Integer(2));
                   l.add(new Integer(3));
                   l.add(new Integer(4));
                   l.add(new Integer(5));
                   return "done";
            }


            public Integer[] getMyArray() {
                   return a;
            }

            public List getMyList() {
                   return l;
             }

      public Decider getMyDecider() {
                return new Decider() {
                        public boolean decide(Object element) throws Exception {
                                int i = ((Integer)element).intValue();
                                return (((i % 2) == 0)?true:false);
                        }
                };
                }
        }
```

```
<!-- A: List basic -->
   <ww:subset source="myList">
```

```
         <ww:iterator>
                <ww:property />
         </ww:iterator>
       </ww:subset>
```

```
    <!-- B: List with count -->
       <ww:subset source="myList" count="3">
                 <ww:iterator>
                         <ww:property />
                 </ww:iterator>
        </ww:subset>
```

```
    <!--  C: List with start -->
         <ww:subset source="myList" count="13" start="3">
                   <ww:iterator>
                         <ww:property />
                   </ww:iterator>
         </ww:subset>
```

```
    <!--  D: List with id -->
         <ww:subset id="mySubset" source="myList" count="13" start="3" />
         <%
                    Iterator i = (Iterator) pageContext.getAttribute("mySubset");
             while(i.hasNext()) {
         %>
         <%=i.next() %>
         <%  } %>
```

```
    <!--  D: List with Decider -->
         <ww:subset source="myList" decider="myDecider">
                       <ww:iterator>
                        <ww:property />
               </ww:iterator>
         </ww:subset>
```

Data Tags

This page last changed on May 10, 2006 by scud.

Data标签用来提供各种数据相关的功能.范围从显示一个action的直接结果,到获取本地化的数值等.

1. action
2. bean
3. debug
4. i18n
5. include
6. param
7. push
8. set
9. text
10. url
11. property

action

This page last changed on May 10, 2006 by scud.

描述

通过指定命名空间和action名称,该标签允许你在jsp页面直接调用Action. 标签体用来渲染Action执行结果. 除非你设定了executeResult参数为true,否则你在xwork.xml中为该Action指定的Result Processor不会执行.

参数

名称	必填	缺省	类型	描述
id	false		String	如果设定,将作为该Action在栈中的标识
name	true		String	action名字(不包括后缀,如.action)
namespace	false		String	action所在命名空间
executeResult	false	false	Boolean	Action的result是否需要被执行
ignoreContextParams	false	false	Boolean	request中的参数是否需要传入该Action

例子

```
public class ActionTagAction extends ActionSupport {

public String execute() throws Exception {
        return "done";
}

public String doDefault() throws Exception {
        ServletActionContext.getRequest().setAttribute("stringByAction", "This is a String put
in by the action's doDefault()");
        return "done";
}
}
```

```
<xwork>
   ....
  <action name="actionTagAction1" class="tmjee.testing.ActionTagAction">
      <result name="done">success.jsp</result>
  </action>
   <action name="actionTagAction2" class="tmjee.testing.ActionTagAction" method="default">
      <result name="done">success.jsp</result>
  </action>
   ....
</xwork>
```

```
<div>The following action tag will execute result and include it in this page</div>
<br />
<ww:action name="actionTagAction" executeResult="true" />
 <br />
 <div>The following action tag will do the same as above, but invokes method specialMethod in
action</div>
<br />
<ww:action name="actionTagAction!specialMethod" executeResult="true" />
 <br />
 <div>The following action tag will not execute result, but put a String in request scope
      under an id "stringByAction" which will be retrieved using property tag</div>
 <ww:action name="actionTagAction!default" executeResult="false" />
 <ww:property value="#attr.stringByAction" />
```

bean

This page last changed on May 10, 2006 by scud.

描述

实例化一个符合JavaBeans规范的class,标签体内可以包含几个Param元素,用于调用setter方法给此class的属性赋值.如果是定了id属性,则该实例将会放到stack的context中.

参数

名称	必填	缺省	类型	描述
name	true		String	要被实例化的class名字(必须符合JavaBeans规范)
id	false		String/Object	标识该元素

例子

译注

原文中此处Freemarker的例子写法格式可能有误.一般的写法应该为类似:
<@ww.bean ... />的方式

```
<-- in freemarker form -->
[ww.bean name="com.opensymphony.webwork.example.counter.SimpleCounter" id="counter"]
  [ww:param name="foo" value="BAR"/]
  The value of foo is : [ww:property value="foo"/], when inside the bean tag.<br />
[/ww:bean]
```

```
<-- in jsp form -->
<ww:bean name="com.opensymphony.webwork.example.counter.SimpleCounter" id="counter">
         <ww:param name="foo" value="BAR" />
  The value of foot is : <ww:property value="foo"/>, when inside the bean tag <br />
</ww:bean>
```

上面的例子初始化了一个叫SimpleCounter的class实例并且给其foo属性赋了值(setFoo('BAR')). 然后该实例被压进Valuestack, 也就是说我们可以通过Property标签调用其getter方法(getFoo())取到相应的值.

上例中, 我们把id设为counter. 导致该实例会被放进stack的context中.你可以通过标签取到该实例

```
<-- jsp form -->
<ww:property value="#counter" />
```

```
<-- freemarker form -->
[ww:property value="#counter.foo"/]
```

上面property标签的例子中,#告诉Ognl去context中找id为counter的SimpleCounter实例.

debug

This page last changed on May 10, 2006 by scud.

debug

名称	必填	缺省	类型	描述
id	string	FALSE		

i18n

This page last changed on May 10, 2006 by scud.

描述

将某个特定resource bundle放入value stack. 然后通过text标签拿到相应message, 而不是仅限于绑定到当前action的bundle.

参数

名称	必填	缺省	类型	描述
name	true		String	要使用的resource bundle.(如foo/bar/customBundle)
id	false		String/Object	标识该元素

例子

```
<ww:i18n name="myCustomBundle">
   The i18n value for key aaa.bbb.ccc in myCustomBundle is <ww:property
value="text('aaa.bbb.ccc')" />
</ww:i18n>
```

include

This page last changed on May 10, 2006 by scud.

描述

包含servlet的输出(servlet或JSP页面)

参数

名称	必填	缺省	类型	描述
value	true		String	包含的jsp或servlet
id	false		Object/String	元素标识

例子

```
<-- One: -->
<ww:include value="myJsp.jsp" />

<-- Two: -->
<ww:include value="myJsp.jsp">
   <ww:param name="param1" value="value2" />
   <ww:param name="param2" value="value2" />
</ww:include>

<-- Three: -->
<ww:include value="myJsp.jsp">
   <ww:param name="param1">value1</ww:param>
   <ww:param name="param2">value2<ww:param>
</ww:include>
```

示例1 - 包含一个 myJsp.jsp 页面
示例2 - 包含一个 myJsp.jsp 页面,附带参数 param1=value1 和 param2=value2
示例3 - 包含一个 myJsp.jsp 页面,附带参数 param1=value1 和 param2=value2

param

This page last changed on May 10, 2006 by scud.

描述

为其他标签提供参数,比如include标签和bean标签.
参数的name属性是可选的,如果提供,会调用Component的方法addParameter(String, Object),如果不提供,则外层嵌套标签必须实现UnnamedParametric接口(如TextTag).

该标签的两个属性

- name (String) - 参数名
- value (Object) - 参数值

注意 : value的提供有两种方式,通过value属性或者标签中间的text,不同之处我们看一下例子:

```
<param name="color">blue</param> <-- (A) -->
<param name="color" value="blue"/> <-- (B) -->
```

(A)中,参数值会以String的格式放入statck. (B)中该值会以java.lang.Object的格式放入statck.
具体请看http://jira.opensymphony.com/browse/WW-808.

参数

名称	必填	缺省	类型	描述
name	false		String	参数名
value	false	从stack中evaluated的值	Object/String	value表达式
id	false		Object/String	引用元素的id

例子

```
<pre>
<ui:component>
 <ui:param name="key"     value="[0]"/>
 <ui:param name="value"   value="[1]"/>
 <ui:param name="context" value="[2]"/>
</ui:component>
</pre>
```

这里key作为标识符,value在当前的OgnlValueStack上被当作一个OGNL的表达式来求值获得结果.(原文:where the key will be the identifier and the value the result of an OGNL expression run against the current OgnlValueStack.)

property

This page last changed on May 10, 2006 by scud.

描述

得到'value'的属性,如果value没提供,默认为堆栈顶端的元素.

参数

名称	必填	缺省	类型	描述
default	false		String	如果属性是null则显示的default值
escape	false	true	Boolean	是否escape HTML
value	false	<top of stack>	Object	value to be displayed
id	false		Object/String	该元素标识

例子

```
<ww:push value="myBean">
    <!-- Example 1: -->
    <ww:property value="myBeanProperty" />

    <!-- Example 2: -->
    <ww:property value="myBeanProperty" default="a default value" />
</ww:push>
```

Example 1 显示出myBean's getMyBeanProperty() 的执行结果.
Example 2 显示出myBean's getMyBeanProperty() 的执行结果,如果是null,则显示缺省(default)值.

push

This page last changed on May 10, 2006 by scud.

描述

push值到堆栈中,方便应用.

参数

名称	必填	缺省	类型	描述
value	true		Object/String	要push到堆栈中的值
id	false		Object/String	该元素标识

例子

```
<ww:push value="user">
    <ww:propery value="firstName" />
    <ww:propery value="lastName" />
</ww:push>

#user##push###,###property#######user###(firstName, lastName etc)
```

```
<ww:push value="myObject">                              ----- (1)
      <ww:bean name="jp.SomeBean" id="myBean"/>         ----- (2)
                    <ww:param name="myParam" value="top"/>        ----- (3)
      </ww:bean>
</ww:push>
```

当在 (1) 时, myObject 在栈顶
当在 (2) 时, jp.SomeBean 在栈顶, 也在stack's context中以myBean为键值存在
当在 (3) 时, top得到jp.SomeBean的实例

```
<ww:push value="myObject">                                       ---(A)
   <ww:bean name="jp.SomeBean" id="myBean"/>                   ---(B)
      <ww:param name="myParam" value="top.mySomeOtherValue"/>  ---(C)
   </ww:bean>
</ww:push>
```

当在 (A) 时, myObject 在栈顶
当在 (B) 时, jp.SomeBean 在栈顶, 也在stack's context中以myBean为键值存在
当在 (C) 时, top指向jp.SomeBean的实例.所以top.mySomeOtherValue调用SomeBean的 mySomeOtherValue() 方法

```
<ww:push value="myObject">                           ---- (i)
   <ww:bean name="jp.SomeBean" id="myBean"/>       ---- (ii)
      <ww:param name="myParam" value="[1].top"/>   -----(iii)
   </ww:bean>
</ww:push>
```

当在（i）时，myObject 在栈顶
当在（ii）时，jp.SomeBean 在栈顶，后面是myObject
当在（iii）时，[1].top返回myObject

set

This page last changed on May 10, 2006 by scud.

描述

set标签赋予变量一个特定范围内的值. 当希望给一个变量赋一个复杂的表达式,每次访问该变量而不是复杂的表达式时用到.其在两种情况下非常有用: 复杂的表达式很耗时 (性能提升) 或者很难理解 (代码可读性提高).

参数

名称	必填	缺省	类型	描述
name	true		String	变量的名字
scope	false	action	String	变量作用域,可以为application, session, request, page, 或action.
value	false		Object/String	将会赋给变量的值
id	false		Object/String	元素标识

例子

```
<ww:set name="personName" value="person.name"/>
Hello, <ww:property value="#personName"/>. How are you?
```

text

This page last changed on May 10, 2006 by scud.

描述

支持国际化信息的标签

国际化信息必须放在一个和当前action同名的resource bundle中,如果没有找到相应message,tag body将被当作默认message,如果没有tag body,message的name会被作为默认message

参数

名称	必填	缺省	类型	描述
name	true		Object/String	资源属性的名字
name	false		Object/String	该元素标识

例子

```
<!-- First Example -->
<ww:i18n name="webwork.action.test.i18n.Shop">
    <ww:text name="main.title"/>
</ww:i18n>

<!-- Second Example -->
<ww:text name="main.title" />
```

另外一个例子

```
<ww:text name="format.money"><ww:param name="value" value="myMoneyValue"/></ww:text>
```

这里

```
format.money={0,number,currency}
```

了解格式化文本的各种情况的文档,请浏览

1. http://java.sun.com/j2se/1.4.2/docs/api/java/text/MessageFormat.html
2. http://java.sun.com/docs/books/tutorial/i18n/format/decimalFormat.html

url

This page last changed on May 10, 2006 by scud.

描述

该标签用于创建url,可以通过"param"标签提供request参数.

注意:
当includeParams的值时'all'或者'get', param标签中定义的参数将有优先权,也就是说其会覆盖其他同名参数的值.

参数

名称	必填	缺省	类型	描述
includeParams	false	get	Object/String	值为'none', 'get'或'all'.
scheme	false		Object/String	scheme属性
value	false		Object/String	value如果不提供就用当前action
action	false		Object/String	用来生成url的action,如果没有则使用value
namespace	false		Object/String	命名空间
method	false		Object/String	使用的action的方法
encode	false	true	Boolean	是否encode参数
includeContext	false	true	Boolean	是否实际的上下文环境应该包含在url中
portletMode	false		Object/String	结果portlet 的模式(mode)
windowState	false		Object/String	结果portlet窗口的状态
portletUrlType	false		Object/String	指定这时一个portlet 输出还是一个 action url
anchor	false		Object/String	URL的锚点(anchor)
id	false		Object/String	该元素标识

例子

```
<-- Example 1 -->
<ww:url value="editGadget.action">
    <ww:param name="id" value="%{selected}" />
</ww:url>
```

```
<-- Example 2 -->
<ww:url action="editGadget">
    <ww:param name="id" value="%{selected}" />
</ww:url>
```

```
<-- Example 3-->
<ww:url includeParams="get"  >
    &lt:param name="id" value="%{'22'}" />
</ww:url>
```

Form Tags

This page last changed on May 21, 2006 by dogg.

描述

对于表单标签,分为两种标签:form标签本身,和所有来包装单个的表单元素的其他标签.form标签本身的行为不同于它内部的元素,这是很重要的.在我们为所有表单标签,包括form标签在内,提供一个参考手册之前,我们必须先描述一些通用的属性.

⚠ 请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

表单标签 Themes

就像前面我们在Themes and Templates里面提到的,HTML标签(包括表单标签)都是模板驱动的.模板按照form的theme分组.WebWork缺省提供了三种theme:

- simple
- xhtml, 扩展了 simple (缺省theme)
- ajax, 扩展了 xhtml

记住: xhtml theme 输出两列表格. 如果你需要不同的布局,我们强烈推荐你 **不要** 编写自己的HTML,而是创建自己的theme或者利用simple theme.

使用simple theme的缺点就是它不支持其他theme那么多的属性.例如.label属性在simple theme里没有任何用处.类似的,simple theme提供的功能也远远少于xhtml和ajax theme提供的:自动显示错误信息就不被支持.

通用属性

所有表单标签扩展了 UIBean 类.这个基类有一些通用属性,分为三种:模板相关的,javascript相关的和通用属性.我们不会在这里说明这些属性,而是维护每个标签的参考.然而,熟悉UI标签的结构以及那些属性对所有标签是可用的,也不失为一个好主意.

除了这些属性之外,对于所有表单元素标签存在一个特殊的属性: form(例如: ${parameters.form}). 这代表输出form标签的参数,并且允许你在你的表单元素和表单(form)本身之间进行交互.例如,在一个模板里,你可以通过${parameters.form.id} 访问form的ID.

模板相关属性

属性	Theme	数据类型	描述
templateDir	n/a	String	定义模板目录
theme	n/a	String	定义theme名称
template	n/a	String	定义模板名称

Javascript相关属性

属性	Theme	数据类型	描述
onclick	simple	String	html javascript onclick 属性
ondbclick	simple	String	html javascript ondbclick 属性
onmousedown	simple	String	html javascript onmousedown 属性
onmouseup	simple	String	html javascript onmouseup 属性
onmouseover	simple	String	html javascript onmouseover 属性
onmouseout	simple	String	html javascript onmouseout 属性
onfocus	simple	String	html javascript onfocus 属性
onblur	simple	String	html javascript onblur 属性
onkeypress	simple	String	html javascript onkeypress 属性
onkeyup	simple	String	html javascript onkeyup 属性
onkeydown	simple	String	html javascript onkeydown 属性
onselect	simple	String	html javascript onselect 属性
onchange	simple	String	html javascript onchange 属性

Tooltip 相关属性

属性	数据类型	缺省值	描述
tooltip	String	none	设置此组件的tooltip
tooltipIcon	String	/webwork/static/tooltip/tooltip.gif	tooltip图标的url
tooltipAboveMousePointer	Boolean	false	在鼠标光标位置上放置tooltip.另外设置tooltipOffseY 允许你设置从鼠标光标位置的垂直位移.
tooltipBgColor	String	#e6ecff	tooltip的背景色.
tooltipBgImg	String	none	背景图片.
tooltipBorderWidth	String	1	tooltip边框的宽度.
tooltipBorderColor	String	#003399	tooltip边框的背景色
tooltipDelay	String	500	显示Tooltip的时间延迟(毫秒). 类似基于操作系统的

			tooltip的行为.
tooltipFixCoordinateX	String	not specified	固定tooltip在指定的X坐标上.例如和tooltipSticky属性结合时很有用.
tooltipFixCoordinateY	String	not specified	固定tooltip在指定的Y坐标上.例如和tooltipSticky属性结合时很有用.
tooltipFontColor	String	#000066	字体颜色.
tooltipFontFace	String	arial,helvetica,sans-serif	字体,例如 verdana,geneva,sans-serif
tooltipFontSize	String	11px	字体大小,例如 30px
tooltipFontWeight	String	normal	Font weight. 可以是 normal 或者 bold
tooltipLeftOfMousePointer	Boolean	false	在鼠标光标位置左侧的Tooltip位置
tooltipOffsetX	String	12	相对鼠标光标位置的水平位移.
tooltipOffsetY	String	15	相对鼠标光标位置的垂直位移.
tooltipOpacity	String	100	tooltip的透明度. 不透明度是行对透明度而言的.设置的值必须是一个介于0(完全透明)和100(不透明)之间的数字.Opera尚未支持.
tooltipPadding	String	3	内部间隔,例如,边框和内容之间的空格,例如文字或者图片
tooltipShadowColor	String	#cccccc	使用指定的颜色创建阴影.
tooltipShadowWidth	String	5	使用指定的宽度(距离)创建阴影.
tooltipStatic	Boolean	false	就像基于操作系统的tooltip, tooltip不随着鼠标光标移动而移动.
tooltipSticky	Boolean	false	tooltip一直停留在它初始的位置,直到另外一个tooltip被激活,或者用户点击了文档.
tooltipStayAppearTime	String	0	指定一个tooltip消失的时间间隔(毫秒),即时鼠标还在相关的HTML元素上不懂,设置<=0,就和没有定义一样.
tooltipTextAlign	String	left	调整包括标题和tooltip内容的文字位置.可以是 right, left 或 justify
tooltipTitle	String	none	标题
tooltipTitleColor	String	#ffffff	title文字的颜色
tooltipWidth	String	300	tooltip的宽度

通用属性

属性	Theme	数据类型	描述
cssClass	simple	String	定义 html class 属性
cssStyle	simple	String	定义html style 属性
title	simple	String	定义html title 属性
disabled	simple	String	定义html disabled 属性
label	xhtml	String	定义表单元素的label
labelPosition	xhtml	String	定义表单元素的label位置(top/left),缺省为left
requiredposition	xhtml	String	定义required 标识相对label元素的位置(left/right),缺省是right
name	simple	String	表单元素的name映射
required	xhtml	Boolean	在label中添加 * (true增加,否则不增加)
tabIndex	simple	String	定义html tabindex 属性
value	simple	Object	定义表单元素的value

什么时候一些属性不起作用(When Some Attributes Don't Apply)

注意有一些标签有一些任何模板都没有使用的属性,可能是没有意义或者是不需要.例如,form标签,支持 tableindex 属性,但是没有一个theme能输出它.同时,就像前面提到的,特定的theme不会利用一些属性.

Value/Name 的关系

在很多标签里,除了form标签,在 name 和 value 属性之间存在一个独特的关系. name 属性用于得到表单元素的名字以及提交时用到.实际上它也是你希望绑定值的表达式.在大多数情况下,它是一个简单的JavaBean属性,例如 "firstName".这会最终调用setFirstName.

类似的,你经常也希望在你的表单元素里显示相同的JavaBean属性的已经存在的数据.现在, value 属性派上了用场. 一个"%{firstName}"会调用getFirstName来在你的表单里显示它,允许用户编辑这个值并重新提交它.

你可以使用下面的代码,它会工作的很好:

```
<@ww.form action="updatePerson">
    <@ww.textfield label="First name" name="firstName" value="%{firstName}"/>
    ...
</@ww.form>
```

然而,因为 name 和 value 的关系经常是可预知的,我们会自动为你处理这些,这样做就可以:

```
<@ww.form action="updatePerson">
    <@ww.textfield label="First name" name="firstName"/>
    ...
</@ww.form>
```

大多数的属性直接使用和属性相同的key暴露给底层的模板,(例如 ${parameters.label}) , value 属性不是这样的. 相反,它可以通过 "nameValue" 主键来访问 (例如 ${parameters.nameValue}),这表示它可能从 name 属性或者是明确地使用 value 属性定义来生成的.

ID Name 设置

所有的表单标签自动为你设置一个 ID. 你可以自由地按照你希望的那样覆盖这个值. ID的设置是这样工作的:

1. 对于form,ID被设定为action的名字.在前一个例子中,ID会设置为"updatePerson".
2. 对于表单元素,ID为设定为 [form's ID]_[element name]

Required 属性

在很多WebWork UI标签上的"required" 属性只有当你开启了客户端校验并且有一个校验器和特定字段关联时缺省才是true.

Tooltip(工具提示)

每个Form UI组件 (在xhtml/css_xhtml或者其他扩展了它们的theme里) 可以有设置给它们的tooltip.Form组件的tooltip相关的属性一旦定义就会设置给所以在它内部创建的表单UI组件,除非表单元素组件自己明确地在tooltip属性里设定来覆盖.

在例子1中,textfield会从包含它的form中继承 tooltipAboveMousePointer 属性.换句话说,尽管它没有定义一个 tooltipAboveMousePointer 属性,它会从包含它的form的属性中继承过来定义为true的属性.

在例子2中,textfield会从包含它的form继承tooltipAboveMousePointer 和 tooltipLeftOfMousePointer 属性,但是 tooltipLeftOfMousePointer 被textfield自己的属性覆盖了.因此,textfield实际上会有一个为定义为true的 tooltipAboveMousePointer 属性,从包含它的form中继承而来,具有一个定义为false的tooltipLeftOfMousePointer 属性,因为textfield自己覆盖了这个定义.

例子 3, 4 和5 显示了不同的设置tooltipConfig属性的方法.

- 例子 3:通过param标签的标签体(body)设置 tooltip配置.
- 例子 4:通过param标签的value属性来设置tooltip配置
- 例子 5:通过component标签的tooltipConfig属性来设置 tooltip 配置

例子1

```
<ww:form
```

```
                            tooltipConfig="#{'tooltipAboveMousePointer':'true',
                               'tooltipBgColor='#eeeeee'}" .... >
    ....
      <ww:textfield label="Customer Name" tooltip="Enter the customer name" .... />
    ....
  </ww:form>
```

例子2

```
 <ww:form
          tooltipConfig="#{'tooltipAboveMousePointer':'true',
                                        'tooltipLeftOfMousePointer':'true'}" ... >
    ....
      <ww:textfield label="Address"
           tooltip="Enter your address"
           tooltipConfig="#{'tooltipLeftOfMousePointer':'false'}" />
    ....
  </ww:form>
```

例子3

```
 <ww:textfield
         label="Customer Name"
               tooltip="One of our customer Details'">
         <ww:param name="tooltipConfig">
              tooltipWidth = 150 |
              tooltipAboveMousePointer = false |
              tooltipLeftOfMousePointer = false
         </ww:param>
  </ww:textfield>
```

例子4

```
 <ww:textfield
                 label="Customer Address"
                 tooltip="Enter The Customer Address" >
                 <ww:param
               name="tooltipConfig"
               value="#{'tooltipStatic':'true',
                        'tooltipSticky':'true',
                        'tooltipAboveMousePointer':'false',
                        'tooltipLeftOfMousePointer':'false'}"  />
  </ww:textfield>
```

例子5

```
 <ww:textfield
           label="Customer Telephone Number"
           tooltip="Enter customer Telephone Number"
           tooltipConfig="#{'tooltipBgColor':'#cccccc',
                            'tooltipFontColor':'#eeeeee',
                            'tooltipAboveMousePointer':'false',
                            'tooltipLeftOfMousePointer':'false'}" />
```

表单标签参考手册

有一点很重要值得注意,所有的插入某些内容到valuestack里的标签(例如i18n或者bean标签)会在标签结束时移除这些对象.这意味着如果你实例化了一个bean使用bean标签 (<ww:bean name="'br.univap.fcc.sgpw.util.FormattersHelper'">) ,这个bean会一直在valuestack里可以使用,直到</ww:bean>标签出现.

1. checkbox - 输出一个复选框
2. checkboxlist - 输出一个复选框列表
3. combobox - 输出一个部件,可以从下拉框的内容填充一个文本框
4. datepicker - 输出一个日期选择不见,使用了 JavaScript 和 DOM
5. doubleselect - 输出一个双选下拉框部件,第二个下拉框依赖第一个
6. head - 输出对应theme的HEAD部分的内容,例如 CSS 和 JavaScript 引用
7. file - 输出一个文件选择框
8. form - 输出一个form(表单)
9. hidden - 输出一个hidden表单字段
10. label - 输出一个 label
11. optiontransferselect- 输出一个选项移动下拉组件,主要是两个下拉框和用来在两个下拉框之间移动选项的按钮.
12. password - 输出一个密码输入框
13. radio - 输出一个单选框
14. reset - 输出一个reset表单按钮
15. richtexteditor - 输出一个富文本编辑器
16. select - 输出一个下拉框
17. submit - 输出一个submit按钮
18. textarea - 输出一个文本输入区域(textarea)
19. textfield - 输出一个文本输入框
20. token - 输出一个隐藏的字段来防止多次提交表单
21. updownselect - 输出一个下拉框组件,带有上下按钮来移动下拉框组件的元素

checkbox

This page last changed on May 05, 2006 by dogg.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

输出一个类型为checkbox的HTML input元素, 使用OGNL值堆栈(OgnlValueStack)中的指定属性配置该元素

属性

名称	必填	缺省值	类型	描述
fieldValue	false	'true'	Object/String	checkbox的真实HTML的value属性.
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex

value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置(有待测试后补充翻译)
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

JSP:

```
<ww:checkbox label="checkbox test" name="checkboxField1" value="aBoolean" fieldValue="true"/>
```

Velocity:

```
#tag( Checkbox "label=checkbox test" "name=checkboxField1" "value=aBoolean" )
```

生成的HTML (simple template, aBoolean == true):

```
<input type="checkbox" name="checkboxField1" value="true" checked="checked" />
```

checkboxlist

This page last changed on May 20, 2006 by scud.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

使用一个列表创建一系列复选框,属性配置与<ww:select /> or <ww:radio />相似,但是创建的是checkbox标签.

属性

名称	必填	缺省值	类型	描述
list	true		Object/String	设置的用来迭代的值.如果list是一个Map(key,value), Map的key会成为选项的"value"的参数,Map的value会成为选项的内容体.
listKey	false		Object/String	list内含对象的用来获取字段的value的属性
listValue	false		Object/String	list内含对象用来获取字段的内容的属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字

required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:checkboxlist name="foo" list="bar"/>
```

combobox

This page last changed on May 22, 2006 by foxcoming.

⚠ 请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

本标签使用一个text类型的INPUT和一个SELECT组合在一起提供组合框(Combo Box)的功能. 可以通过使用SELECT控件将文本输入到INPUT控件中, 或直接在文本框中输入文本.

本例中, SELECT使用id=year属性生成. Counter本身就是一个Iterator. 他将从first(值为text('firstBirthYear'))枚举到last(值为2000). SELECT是使用JavaScript生成的, 因此该标签应当包含在<form>标签中.

需要注意的是, 与<ww:select/>标签不同, 不能分别定义每个<option>标签的id(应该为value, 译注)属性或文本内容. 每一个属性都是直接使用列表元素的toString()方法生成的. 这大概是因为这里的下拉列表不关心实际提交的数据(指option的value属性, 译注), 而仅仅为了帮助用户填写文本框而已.

属性

名称	必填	缺省值	类型	描述
list	**true**		Object/String	生成列表项的可迭代数据源. 如果没有指定该属性, 将不显示下拉列表控件.
maxlength	false		Integer	HTML maxlength属性
maxLength	false		Object/String	不建议使用. 建议使用maxlength属性替代.
readonly	false	false	Boolean	设置为只读,不允许输入
size	false		Integer	HTML size 属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled

label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)

tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

(snippet:id=tagattributes|javadoc=false|url=webwork/docs/tags/ComboBox.html)

示例

```
JSP:
<ww:bean name="webwork.util.Counter" id="year">
  <ww:param name="first" value="text('firstBirthYear')"/>
  <ww:param name="last" value="2000"/>

  <ww:combobox label="Birth year" size="6" maxlength="4" name="birthYear" list="#year"/>
</ww:bean>

Velocity:
#tag( ComboBox "label=Birth year" "size=6" "maxlength=4" "name=birthYear" "list=#year" )
```

datepicker

This page last changed on May 23, 2006 by foxcoming.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

绘制日期选取器控件.

WebWork 2.2版本的实现使用jscalendar替换了不支持本地化的tigracalendar. 如果在应用中使用了旧版本的控件, 请检查地区和日期格式设置. 如果不想让日历看起来是透明的, 确保在页面中包含了下面列出的正确的样式表(stylesheet).

重要: 如果不在<ww:form />标签中使用时, 一定要设置id属性, 将选中的日期值复制到文本框元素时需要使用该属性.

下面是format参数的参考(从jscalendar的文档中复制):

%a	简写的星期名称 (三个英文, 或一个中文)
%A	完整的星期名称
%b	简写的月份名称 (三位的字母或中文数字)
%B	完整的月份名称
%C	显示世纪
%d	两位数的日期(00 .. 31)
%e	日期 (0 .. 31)
%H	小时 (00 .. 23)
%I	小时 (01 .. 12)
%j	在一年中的天数序号(000 .. 366)
%k	小时 (0 .. 23)
%l	小时 (1 .. 12)
%m	数字月份 (01 .. 12)
%M	分钟数 (00 .. 59)
%n	换行符
%p	``PM''或``AM''
%P	``pm''或``am''
%S	秒(00 .. 59)
%s	从计时点开始的描述 (从1970年1月1日零点开始)
%t	制表符 ('\t')
%U, %W, %V	在一年中的星期序号 (如: 第一周)
%u	在星期中的序号 (1 .. 7, 1 = MON)
%w	在星期中的序号 (0 .. 6, 0 = SUN)
%y	不包含世纪的年份 (00 .. 99)
%Y	包含世纪的年份 (ex. 1979)

%%	一个百分号('%')

(摘自snippet:id=javadoc|javadoc=true|url=com.opensymphony.webwork.components.DatePicker)

属性

名称	必填	缺省值	类型	描述
language	false	当前地区使用的语言	String	控件文本使用的语言和本地化设置.
format	false	由语言指定的预设的日期格式(英语("en")使用%Y/%m/%d)	String	日期字段的格式.
showstime	false	false	String	是否显示时间选取器.有效值包括"true", "false", "24"和"12".
singleclick	false	true	Boolean	决定使用单击选中还是双击选中.
maxlength	false		Integer	HTML maxlength属性
maxLength	false		Object/String	不建议使用. 建议使用maxlength属性替代.
readonly	false	false	Boolean	设置为只读,不允许输入
size	false		Integer	HTML size 属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)

name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
Date in application's locale format:
    <ww:datepicker name="order.date" id="order.date" />
Date in german locale, with german texts:
    <ww:datepicker name="delivery.date" id="delivery.date" template="datepicker_js.ftl"
language="de" />
Date in german locale, with german texts and custom date format, including time:
    <ww:datepicker name="invoice.date" id="invoice.date" template="datepicker_js.ftl"
language="de" format="%d. %b &Y %H:%M" showstime="true" />
```

要使用这个基于jscalendar的日期选取器控件, 需要使用jscalendar提供的标准样式表(全部发行的样式表都包含在webwork的jar文件中). 最简单的做法是在HTML页面的加入<ww:head/>标签, 该标签会包含jscalendar的css文件. 否则, 要手工激活calendar-blue风格, 需要在css引用中包含下列内容:

```
<link href="<ww:url value="/webwork/jscalendar/calendar-blue.css" />" rel="stylesheet"
type="text/css" media="all"/>
```

doubleselect

This page last changed on May 24, 2006 by foxcoming.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

绘制两个HTML Select元素, 第二个列表显示的内容随第一个列表的选中的选项变化.

属性

名称	必填	缺省值	类型	描述
doubleList	true		Object/String	创建第二个列表的可迭代数据源.
doubleListKey	false		Object/String	第二个列表使用的主键表达式
doubleListValue	false		Object/String	第二个列表使用的选项值表达式
doubleName	true		Object/String	整个组件的名称
doubleValue	false		Object/String	整个组件的值表达式
formName	false		Object/String	组件所在的form名称
doubleCssClass	false		Object/String	第二个列表的class
doubleCssStyle	false		Object/String	第二个列表的CSS风格
doubleHeaderKey	false		Object/String	第二个列表的题头主键值
doubleHeaderValue	false		Object/String	第二个列表的题头选项值
doubleEmptyOption	false		Object/String	第二个列表是否添加空选项
doubleDisabled	false		Object/String	第二个列表是否可以使用disable属性
doubleId	false		Object/String	第二个列表的id
doubleMultiple	false		Object/String	是否设置第二个列表的multiple属性
doubleOnblur	false		Object/String	设置第二个列表的onblur属性
doubleOnchange	false		Object/String	设置第二个列表的onchange属性
doubleOnclick	false		Object/String	设置第二个列表的onclick属性
doubleOndblclick	false		Object/String	设置第二个列表的ondbclick属性
doubleOnfocus	false		Object/String	设置第二个列表的

				onfocus属性
doubleOnkeydown	false		Object/String	设置第二个列表的onkeydown属性
doubleOnkeypress	false		Object/String	设置第二个列表的onkeyperss属性
doubleOnkeyup	false		Object/String	设置第二个列表的onkeyup属性
doubleOnmousedown	false		Object/String	设置第二个列表的onmousedown属性
doubleOnmousemove	false		Object/String	设置第二个列表的onmousemove属性
doubleOnmouseout	false		Object/String	设置第二个列表的onmouseout属性
doubleOnmouseover	false		Object/String	设置第二个列表的onmouseover属性
doubleOnmouseup	false		Object/String	设置第二个列表的onmouseup属性
doubleOnselect	false		Object/String	设置第二个列表的onselect属性
doubleSize	false		Object/String	设置第二个列表的size属性
emptyOption	false	false	Boolean	第一个列表是否添加空选项
headerKey	false		Object/String	设置第一个列表的题头主键值. 一定不能为空值! "'-1'"或"''"是正确的取值, ""是错误的取值.
headerValue	false		Object/String	第一个列表的题头选项值
multiple	false		Object/String	创建一个多选列表. 如果value属性指定了一个数组(正确的元素类型), 那么将预先选中数组中指定的多个选项.
size	false		Integer	该组件列表框的大小(显示元素的个数)
list	true		Object/String	创建列表的可迭代数据源. 如果该列表是一个Map(key, value), 那么Map的主键将作为选项(<option>)的"value"属性, 而该主键对应的值作为选项的文本内容.
listKey	false		Object/String	列表数据源中元素对象的属性, 用于获取选项的值
listValue	false		Object/String	列表数据源中元素对象的属性, 用于获取

				选项的文本内容
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置

				html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:doubleselect label="doubleselect test1" name="menu" list="{'fruit','other'}"
doubleName="dishes" doubleList="top == 'fruit' ? {'apple', 'orange'} : {'monkey', 'chicken'}"
/>
<ww:doubleselect label="doubleselect test2" name="menu" list="#{'fruit':'Nice Fruits',
'other':'Other Dishes'}" doubleName="dishes" doubleList="top == 'fruit' ? {'apple', 'orange'} :
{'monkey', 'chicken'}" />
```

file

This page last changed on May 20, 2006 by scud.

⚠ 请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

绘制一个HTML file input元素

属性

名称	必填	缺省值	类型	描述
accept	false		Object/String	HTML accept 属性,用来标识接收的文件类型(minetype)
size	false		Integer	HTML size 属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex

value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:file name="anUploadFile" accept="text/*" />
<ww:file name="anohterUploadFIle" accept="text/html,text/plain" />
```

form

This page last changed on May 20, 2006 by scud.

请确认你已经读过了Tag Syntax文档,并且理解标签的属性语法是如何工作的.

描述

输出一个HTML输入表单.
远程表单(remote form)允许不刷新页面提交表单.结果表单(results from)能在表单当前页插入任意的html元素.

属性

名称	必填	缺省值	类型	描述
onsubmit	false		Object/String	HTML onsubmit 属性
action	false	current action	Object/String	设置提交的action名字, 不需要 .action 后缀
target	false		Object/String	HTML表单target 属性
enctype	false		Object/String	HTML表单enctype 属性
method	false		Object/String	HTML表单method 属性
namespace	false	current namespace	Object/String	所提交Action的命名空间
validate	false	false	Boolean	是否执行客户端/远程校验. 使用xhtml/ajax或者继承它们的theme时有效
portletMode	false		Object/String	表单提交后显示的portlet 模式
windowState	false		Object/String	表单提交后 要显示的window 的 state
openTemplate	false		Object/String	用来输出html开始部分的模板.
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title

disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属

				性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:form ... />
```

校验

有两种形式的 客户端校验 , 依赖于你对主题(theme)的设置(xhtml, ajax, 等等). 如果你设置为xhtml主题 或 css xhtml主题 , 将会是纯客户端校验. 如果你设置为 ajax主题, 将会是基于AJAX的校验. 请阅读 客户端校验 文档获取更多信息.

head

This page last changed on May 23, 2006 by foxcoming.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

本标签绘制HTML文件中的HEAD部分内容. 因为有些主题需要包含特定的CSS和JavaScript, 因此使用head标签很有用(由于WebWork的标签支持模版, 因此, 不同的主题可以通过修改模版可以很容易的进行调整, 译注).

例如: 如果页面中集成了ajax组件, 那么不需要将缺省主题设置为ajax, 使用一个theme="ajax"的head标签就可以将标准ajax头信息包含在页面中.

本标签也包含用于设置定制的日期选取器主题的选项. 详细描述参见参数calendarcss.

使用ajax主题时可以将debug参数设置为true来打开调试标志.

(摘自snippet:id=javadoc|javadoc=true|url=com.opensymphony.webwork.components.Head)

属性

名称	必填	缺省值	类型	描述
calendarcss	false	Object/String	jscalendar使用的css主题, 缺省为"calendar-blue.css"	
debug	false	Object/String	设置为true将开启AJAX主题的调试模式	
maxlength	false		Integer	HTML maxlength属性
maxLength	false		Object/String	不建议使用. 建议使用maxlength属性替代.
readonly	false	false	Boolean	设置为只读,不允许输入
size	false		Integer	HTML size 属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)

title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置

				html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<head>
  <title>My page</title>
  <ww:head/>
</head>
```

```
<head>
  <title>My page</title>
  <ww:head theme="ajax" calendarcss="calendar-green"/>
</head>
```

hidden

This page last changed on May 05, 2006 by dogg.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

输出一个类型为hidden的HTML input元素,使用OGNL值堆栈(OgnlValueStack)中的指定属性配置该元素.

参数

名称	必填	缺省值	类型	描述
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.

onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<-- ### -->
<ww:hidden name="foo" />
```

```
<-- ### -->
<ww:hidden name="foo" value="bar" />
```

例子一生成的HTML代码 (如果对foo求值为bar):

```
<input type="hidden" name="foo" value="bar" />
```

例子二生成的HTML代码（如果Action的getBar方法返回值为'bar'）

```
<input type="hidden" name="foo" value="bar" />
```

label

This page last changed on May 21, 2006 by dogg.

请确认你已经读过了Tag Syntax文档,并且理解标签的属性语法是如何工作的.

描述

和其他的UI组件一样,一个HTML LABEL 将得到形式为label:name的输出.

属性

名称	必填	缺省值	类型	描述
for	false		Object/String	HTML for 属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.

onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:label label="%{text('user_name')}" name="userName" />
```

在此例子中,将输出一个label,此label从ResourceBundle中调用ActionSupport的getText()方法给你一个输出'User Name:tm_jee'.
前提是假定i18n 消息中 user_name 键相应的值为'User Name',并且Action的getUserName() 方法返回'tm_jee'.

optiontransferselect

This page last changed on May 24, 2006 by foxcoming.

⚠ 请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

创建一个选项转移列表组件, 由两个<select ...>标签以及其间的用于将选项在两个<select ...>之间相互移动的按钮. 表单提交时会自动选中全部选项.

注意: id和doubleId并不需要提供, 因为他们将由包含在form标签中的optiontransferselect标签自动生成. 生成的id和doubleId分别为<form_id>_<optiontransferselect_doubleName>和<form_id>_<optiontransferselect_doubleName>.

属性

名称	必填	缺省值	类型	描述
addAllToLeftLabel	false		Object/String	设置全部移动到左边的按钮的文字
addAllToRightLabel	false		Object/String	设置全部移动到右边的按钮的文字
addToLeftLabel	false		Object/String	设置向左移动的按钮的文字
addToRightLabel	false		Object/String	设置向右移动的按钮的文字
allowAddAllToLeft	false		Object/String	是否使用全部移动到左边的按钮
allowAddAllToRight	false		Object/String	是否使用全部移动到右边的按钮
allowAddToLeft	false		Object/String	是否使用移动到左边的按钮
allowAddToRight	false		Object/String	是否使用移动到右边的按钮
leftTitle	false		Object/String	设置左边列表框的标题
rightTitle	false		Object/String	设置右边列表框的标题
allowSelectAll	false		Object/String	是否使用全部选择按钮
selectAllLabel	false		Object/String	设置全部选择按钮的文字
buttonCssClass	false		Object/String	设置按钮使用的class
buttonCssStyle	false		Object/String	设置按钮使用的风格
doubleList	true		Object/String	创建第二个列表的可迭代数据源.
doubleListKey	false		Object/String	第二个列表使用的主

				键表达式
doubleListValue	false		Object/String	第二个列表使用的选项值表达式
doubleName	true		Object/String	整个组件的名称
doubleValue	false		Object/String	整个组件的值表达式
formName	false		Object/String	组件所在的form名称
doubleCssClass	false		Object/String	第二个列表的class
doubleCssStyle	false		Object/String	第二个列表的CSS风格
doubleHeaderKey	false		Object/String	第二个列表的题头主键值
doubleHeaderValue	false		Object/String	第二个列表的题头选项值
doubleEmptyOption	false		Object/String	第二个列表是否添加空选项
doubleDisabled	false		Object/String	第二个列表是否可以使用disable属性
doubleId	false		Object/String	第二个列表的id
doubleMultiple	false		Object/String	是否设置第二个列表的multiple属性
doubleOnblur	false		Object/String	设置第二个列表的onblur属性
doubleOnchange	false		Object/String	设置第二个列表的onchange属性
doubleOnclick	false		Object/String	设置第二个列表的onclick属性
doubleOndblclick	false		Object/String	设置第二个列表的ondbclick属性
doubleOnfocus	false		Object/String	设置第二个列表的onfocus属性
doubleOnkeydown	false		Object/String	设置第二个列表的onkeydown属性
doubleOnkeypress	false		Object/String	设置第二个列表的onkeyperss属性
doubleOnkeyup	false		Object/String	设置第二个列表的onkeyup属性
doubleOnmousedown	false		Object/String	设置第二个列表的onmousedown属性
doubleOnmousemove	false		Object/String	设置第二个列表的onmousemove属性
doubleOnmouseout	false		Object/String	设置第二个列表的onmouseout属性
doubleOnmouseover	false		Object/String	设置第二个列表的onmouseover属性
doubleOnmouseup	false		Object/String	设置第二个列表的onmouseup属性
doubleOnselect	false		Object/String	设置第二个列表的onselect属性

doubleSize	false		Object/String	设置第二个列表的size属性
emptyOption	false	false	Boolean	第一个列表是否添加空选项
headerKey	false		Object/String	设置第一个列表的题头主键值. 一定不能为空值! "'-1'"或"''"是正确的取值, ""是错误的取值.
headerValue	false		Object/String	第一个列表的题头选项值
multiple	false		Object/String	创建一个多选列表. 如果value属性指定了一个数组(正确的元素类型), 那么将预先选中数组中指定的多个选项.
size	false		Integer	该组件列表框的大小(显示元素的个数)
list	true		Object/String	创建列表的可迭代数据源. 如果该列表是一个Map(key, value), 那么Map的主键将作为选项(<option>)的"value"属性, 而该主键对应的值作为选项的文本内容.
listKey	false		Object/String	列表数据源中元素对象的属性, 用于获取选项的值
listValue	false		Object/String	列表数据源中元素对象的属性, 用于获取选项的文本内容
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.

labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标

				签它会被用作HTML的id属性

例子

```
<-- minimum configuration -->
<ww:optiontransferselect
        label="Favourite Cartoons Characters"
        name="leftSideCartoonCharacters"
        list="{'Popeye', 'He-Man', 'Spiderman'}"
        doubleName="rightSideCartoonCharacters"
        doubleList="{'Superman', 'Mickey Mouse', 'Donald Duck'}"
/>

 <-- possible configuration -->
 <ww:optiontransferselect
        label="Favourite Cartoons Characters"
        name="leftSideCartoonCharacters"
        leftTitle="Left Title"
        rightTitle="Right Title"
        list="{'Popeye', 'He-Man', 'Spiderman'}"
        multiple="true"
        headerKey="headerKey"
        headerValue="--- Please Select ---"
        emptyOption="true"
        doubleList="{'Superman', 'Mickey Mouse', 'Donald Duck'}"
        doubleName="rightSideCartoonCharacters"
        doubleHeaderKey="doubleHeaderKey"
        doubleHeaderValue="--- Please Select ---"
        doubleEmptyOption="true"
        doubleMultiple="true"
/>
```

password

This page last changed on May 23, 2006 by scud.

⚠ 请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

输出一个类型为password的HTML input元素.

属性

名称	必填	缺省值	类型	描述
showPassword	false	false	Boolean	是否显示密码
show	false		Object/String	不建议使用. 建议使用showPassword属性替代.
maxlength	false		Integer	HTML maxlength属性
maxLength	false		Object/String	不建议使用. 建议使用maxlength属性替代.
readonly	false	false	Boolean	设置为只读,不允许输入
size	false		Integer	HTML size属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置

				(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

在此例子中,将会显示一个密码输入栏,对于它的标签属性,我们将调用ActionSupport的getText()方法从资源文件中检索此密码输入栏的标签.

```
<ww:password label="%{text('password')}" name="password" size="10" maxlength="15" />
```

radio

This page last changed on May 25, 2006 by scud.

请确认你已经读过 Tag Syntax 文档并且已经理解标签的属性语法是如何工作的.

描述

输出单选框字段.

参数

名称	必填	缺省值	类型	描述
list	true		Object/String	设置的用来迭代的值. 如果list是一个Map(key,value), Map的key会成为选项的"value"的参数,Map的value会成为选项的内容体.
listKey	false		Object/String	list内含对象的用来获取字段的value的属性
listValue	false		Object/String	list内含对象用来获取字段的内容的属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字

required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

在这里例子里,显示了一个有一系列性别的单选控件.这个性别列表是从属性 id=genders上构建而来的.WW调用了会返回一个Map的getGenders().关于使用listKey和listValue属性的例子,可以查看select标签部分.

本例中, 缺省选中的选项通过调用action类的getMale()方法的返回值确定, 该方法返回值应当等于getGenders()方法返回的map中的某个主键, 这样, 对应的gender就会被选中.

```
<ww:action name="GenderMap" id="genders"/>
<ww:radio label="Gender" name="male" list="#genders.genders"/>
```

reset

This page last changed on May 22, 2006 by scud.

请确认你已经读过了Tag Syntax文档,并且理解标签的属性语法是如何工作的.

描述

提供了一个reset按钮. reset标签与form标签一起使用,用来复位表单.reset有两种不同的输出方式:

- input: 输出的html为 <input type="reset"...>
- button: 输出的html为 <button type="reset"...>

注意button type可以增加分离提交的值和按钮上显示的文字的可能,但是在微软的Internet Explorer上有问题(一直到6.0版本).

属性

名称	必填	缺省值	类型	描述
label	false		Object/String	提供给reset按钮文字,不同于reset的value. 当使用 input类型的reset时没有任何效果,因为此时按钮文字总是采用value参数的值.
action	false		String	设置 action 属性.
method	false		String	设置 method 属性.
align	false		String	HTML align 属性.
type	false	input	String	按钮使用的类型. 有效的值分别是 input, button and image.
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对

				应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配

				置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

例子 1

```
<ww:reset value="%{'Reset'}" />
```

例子 2

```
Render an button reset:
<ww:reset type="button" value="%{'Reset'}" label="Reset the form"/>
```

richtexteditor

This page last changed on May 23, 2006 by foxcoming.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

生成一个基于FCK编辑器(http://www.fckeditor.net)的富文本编辑器.

属性

名称	必填	缺省值	类型	描述
checkBrowser	false	true	Boolean	编辑器在绘制工具条时是否检查浏览器的兼容性
displayError	false	true	Boolean	当绘制编辑器失败时是否显示错误信息.
autoDetectLanguage	false	true	Boolean	通知编辑器自动检测用户的语言设置并调整界面语言. 使用Internet Explorer时，使用Windows控制面板中定义的语言. 使用Firefox时, 使用浏览器自己的语言.
baseHref	false		String	处理链接(包括images, links, styles等元素中的链接)时使用的Base URL. 例如, 如果BaseHref设置为'http://www.fredck.com', 那么一个指向"/images/Logo.gif"的图片地址将被编辑器解释为"http://www.fredck.com/images/Log 而不需要修改image元素的"src"属性.
basePath	false	/webwork/static/richtexteditor/	String	设置服务器上包含FCK编辑器文件的目录
contentLangDirection	false	ltr	String	设置编辑区内容的排列方向, 从左到右(ltr)或从右到左(rtl)
customConfigurationsPath	false		String	设置一个定制的配置文件路径, 可以覆盖某些缺省配置. 推荐使用绝对路径(以"/"开始), 如"/myfckconfig.js".

debug	false	false	Boolean	调用FCKDebug.Output()函数时显示调试窗口.
defaultLanguage	false	en	String	设置编辑器界面上使用的缺省语言. 当AutoDetectLanguage选项关闭或用户语言不可用时使用该属性.
enableSourceXHTML	false	true	String	通知编辑器从所见即所得视图切换成源代码视图时将HTML代码处理成XHTML.
enableXHTML	false	true	String	通知编辑器在提交表单时将HTML代码处理成XHTML.
fillEmptyBlocks	false	true	String	强制在块元素(Block elements, 如P, DIV, H1, PRE等)中加入一个 . 在浏览时, 空的块元素将被"收起(collapsed)", 因此一个空的<p></p>不会显示出来. 在编辑时, 编辑器将"展开(expend)"空的块元素, 这样就可以在其中插入内容了. 该选项设置为"true"可以在编辑和浏览时得到一致的输出结果.
flashBrowserURL	false	/webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?Type=Flash&Connector=connectors/jsp/connector.action	String	用户在"Flash"对话框中点击'Browse Server'按钮时调用的页面. 使用这种办法, 可以创建自己定制的Flash浏览器以便更好的与系统集成在一起.
flashUploadURL	false	/webwork/static/richtexteditor/editor/filemanager/upload/uploader.action?Type=Flash	String	用户在"Flash"对话框中点'Send it to server'按钮时调用的文件上传处理器的URL. 使用这种办法, 可以创建自己定制的Flash上传处理器以便更好的与系统集成在一起.
fontColors	false	000000, 993300, 333300, 003300, 003366, 000080, 333399, 333333, 800000, FF6600, 808000, 808080, 008080, 0000FF, 666699, 808080, FF0000, FF9900, 99CC00, 339966, 33CCCC, 3366FF, 800080,	string	必须显示在颜色面板(在工具条上)的颜色.

		999999, FF00FF, FFCC00, FFFF00, 00FF00, 00FFFF, 00CCFF, 993366, C0C0C0, FF99CC, FFCC99, FFFF99, CCFFCC, CCFFFF, 99CCFF, CC99FF, FFFFFF		
fontFormats	false	p; div; pre; address; h1; h2; h3; h4; h5; h6	string	工具条上的"Format"命令显示的元素列表.
fontNames	false	Arial; Comic Sans MS; Courier New; Tahoma; Times New Roman; Verdana	string	工具条上的"Fonts"命令显示的字体列表.
fontSizes	false	1/xx-small; 2/x-small; 3/small; 4/medium; 5/large; 6/x-large; 7/xx-large	string	工具条上的"Size"命令显示的字体大小列表.
forcePasteAsPlainText	false	false	boolean	再粘贴(paste)操作时将剪贴板的内容强制转换成纯文本.
forceSimpleAmpersand	false	false	boolean	在标签属性值中强制使用'&'符号(不转换成'&')，这种转换是W3C XHTML的要求，因此建议将该选项设置为'false'.
formatIndentator	false	' '	boolean	设置格式化HTML代码时使用的缩进字符. 可以是一系列空格(' ')或一个制表符('\t').
formatOutput	false	true	boolean	处理并格式化编辑器生成的HTML代码.
formatSource	false	true	boolean	当从所见即所得视图切换到源代码视图时，编辑器显示的HTML代码被处理并格式化.
fullPage	false	false	boolean	开启整页编辑功能(从<HTML>起始标签到</HTML>结束). 这也将开启工具条上的'Page Properties'按钮.
geckoUseSPAN	false	true	boolean	通知Gecko浏览器使用SPAN标签来显示粗体，斜体和下划线而不使用标签<B>，<I>和<U>.
height	false	200	string	设置编辑器的高度
imageBrowserURL	false	/webwork/static/richtexteditor/editor/filemanager/browser/default/	string	用户在'Image'对话框中点击'Browse

		browser.html?Type=Image&Connector=connectors/jsp/connector.action		Server'按钮时调用的页面URL. 使用这种办法, 可以创建自己定制的图片浏览器以便更好的与系统集成在一起.
imageUploadURL	false	/webwork/static/richtexteditor/editor/filemanager/upload/uploader.action?Type=Image	string	用户在'Image'对话框中点击'Send it to server'按钮时调用的图片上传处理器. 使用这种办法, 可以创建自己定制的图片上传处理器以便更好的与系统集成在一起.
linkBrowserURL	false	/webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?Type=File&Connector=connectors/jsp/connector.action	string	用户在'Link'对话框中点击'Browse Server'按钮时调用的页面. 使用这种办法, 可以创建自己定制的文件浏览器以便更好的与系统集成在一起.
linkUploadURL	false	*/webwork/static/richtexteditor/editor/filemanager/upload/uploader.action?Type=File *	string	用户在'Link'对话框中点击'Send it to server'按钮时调用的链接上传处理器. 使用这种办法, 可以创建自己定制的链接上传处理器以便更好的与系统集成在一起.
pluginsPath	false	/webwork/static/richtexteditor/plugins/	string	设置插件目录, 编辑器将在该目录下查找注册的插件.
skinPath	false	/webwork/static/richtexteditor/skins/default	string	设置编辑器使用的皮肤(图形界面的设置).
startupFocus	false	false	boolean	强制编辑器在启动时(page load事件)获取键盘输入焦点
stylesXmlPath	false	/webwork/static/richtexteditor/fckstyles.xml	string	设置工具条上'Style'命令使用的定义风格规则的XML文件路径.
tabSpaces	false	0	string	设置当用户输入'tab'键时插入的空格(&nbsp)个数. 这是Internet Explorer特有功能. 其他浏览器自动插入空格.
toolbarCanCollapse	false	true	boolean	当用户点击工具条左边(contentLangDirection为'rtl'时在右边)的竖条时, 编辑器可以收起/展开工具条(on the right for 'rtl' languages).
toolbarSet	false	Default	string	设置工具条的显示名称
toolbarStartExpanded	false	true	boolean	编辑器加载后, 工具

				条是否是展开的.
useBROnCarriageReturn	false	true	boolean	输入回车时是否使用<br/>替换.
width	false	100%	string	设置编辑器的宽度
allowFlashBrowse	false	true	boolean	是否允许flash浏览
allowFlashUpload	false	true	boolean	是否允许flash上传
allowImageBrowse	false	true	boolean	是否允许图片浏览
allowImageUpload	false	true	boolean	是否允许图片上传
allowLinkBrowse	false	true	boolean	是否允许链接浏览
allowLinkUpload	false	true	boolean	是否允许链接上传
flashUploadAllowedExtension	false	.(swf|fla)$	string	允许上传的Flash文件名的正则表达式
flashUploadDeniedExtension	false		string	禁止上传的Flash文件名的正则表达式
imageUploadAllowedExtension	false	.(jpg|gif|jpeg|png)$	string	允许上传的图片文件名的正则表达式
imageUploadDeniedExtension	false		string	禁止上传的图片文件名的正则表达式
linkUploadAllowedExtension	false		string	允许上传的链接名称的正则表达式
linkUploadDeniedExtension	false	.(php| php3| php5| phtml| asp| aspx| ascx| jsp| cfm| cfc| pl| bat| exe| dll| reg| cgi)$	string	禁止上传的链接名称的正则表达式
smileyImages	false		Object/String	包含笑脸图片JavaScript数组的文件.
smileyPath	false	/webwork/static/richtexteditor/editor/images/smiley/msn/	string	笑脸文件所在路径
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式

labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置

id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:richtexteditor
                toolbarCanCollapse="false"
                width="700"
                label="Description 1"
                name="description1"
                value="Some Content I keyed In In The Tag Itself"
                />
```

浏览服务器端内容

有时需要允许一个富文本编辑器浏览服务器端的内容, 例如: 当用户点击图片按钮后, 又在对话框中点击了'Browser'按钮. 为了将该功能与WebWork集成在一起, 需要定义如下所示的action(通常定义在xwork.xml中)

```
<package name="richtexteditor-browse" extends="webwork-default"
  namespace="/webwork/richtexteditor/editor/filemanager/browser/default/connectors/jsp">
        <action name="connector"
     class="com.opensymphony.webwork.components.DefaultRichtexteditorConnector"
     method="browse">
                <result name="getFolders" type="richtexteditorGetFolders" />
                <result name="getFoldersAndFiles" type="richtexteditorGetFoldersAndFiles" />
                <result name="createFolder" type="richtexteditorCreateFolder" />
                <result name="fileUpload" type="richtexteditorFileUpload" />
        </action>
  </package>
```

缺省情况下, 当触发了一个"浏览"命令(例如, 点击'image' button然后在对话框中点击'browse server'按钮, 将触发URL '/webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?&Type=Image&Connector=connectors/jsp/cor
FCK编辑器附带的页面browser.html将触发 '/webwork/richtexteditor/editor/filemanager/browser/default/connectors/jsp/connector.action', 这将执行webwork的*DefaultRichtexteditorConnector*. 通过修改'imageBrowseURL'属性可以改变触发的URL. 一共有三个类似的URL, 分别是'imageBrowseURL', 'linkBrowseURL'和'flashBrowseURL'. 推荐使用缺省的URL. 也可以修改Connector的参数. 例如:

```
/webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?
&Type=Image&Connector=connectors/jsp/connector.action
```

可以修改为:

```
/webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?
&Type=Image&Connector=myLittlePath/myConnector.action
```

本例中的action应当使用名空间'/webwork/richtexteditor/editor/filemanager/browser/default/myLittlePath', 名称应为'myConnector'.

缺省情况下action的方法应该定义在xwork.xml并命名为'browse'. 如果有变化如, myBrowse, 需要使用下列代码:

```
public String myBrowse() {
      browse();
  }
```

(摘自snippet:id=serversidebrowsing|javadoc=true|url=com.opensymphony.webwork.components.RichTextEditor)

上传文件(Server Side Uploading)

有时同样需要允许一个富文本编辑器向服务器上传文件. 例如, 当用户点击图片按钮后, 又在对话框中点击了'Upload'按钮并从客户端机器上选中了一个文件, 然后点击'Send it to the server'. 为了将该功能与WebWork集成在一起, 需要定义如下所示的action(通常定义在xwork.xml中)

```
<package name="richtexteditor-upload" extends="webwork-default"
  namespace="/webwork/richtexteditor/editor/filemanager/upload">
        <action name="uploader"
      class="com.opensymphony.webwork.components.DefaultRichtexteditorConnector"
      method="upload">
                <result name="richtexteditorFileUpload" />
        </action>
  </package>
```

缺省情况下, 当触发了一个"上传"命令时, 将触发URL '/webwork/static/richtexteditor/editor/filemanager/upload/uploader.action?Type=Image'. 通过修改'imageUploadURL'属性可以改变触发的URL. 当链接触发后, 将执行WebWork的action. 一共有三个这样的URL, 名为'imageUploadURL', 'linkUploadURL' and 'flashUploadURL'. 推荐使用缺省的URL. 当然可以修改URL, 但必须包含Type参数. 例如:

```
/webwork/static/richtexteditor/editor/filemanager/upload/uploader.action?Type=Image
```

可以修改为:

```
/webwork/static/richtexteditor/editor/filemanager/upload/aDifferentUploader.action?Type=Image
```

本例中的action应当使用名空间'/webwork/static/richtexteditor/editor/filemanager/upload' , 名称应为'aDifferentUploader'.

缺省情况下action的方法应该定义在xwork.xml并命名为'upload'. 如果有变化如, myUpload, 需要使用下列代码:

```
public String myUpload() {
      upload();
  }
```

(摘自snippet:id=serversideuploading|javadoc=true|url=com.opensymphony.webwork.components.RichTextEditor)

WebWork Action

AbstractRichtexteditorConnector

富文本编辑器用来执行服务器侧浏览和上传的抽象基类.

处理服务器侧浏览和上传的action需要从AbstractRichtexteditorConnector继承.

着需要实现4个抽象方法:

```
 protected abstract String calculateServerPath(String serverPath, String folderPath,
        String type) throws Exception;
  protected abstract Folder[] getFolders(String virtualFolderPath, String type)
        throws Exception;
  protected abstract FoldersAndFiles getFoldersAndFiles(String virtualFolderPath,
        String type) throws Exception;
  protected abstract CreateFolderResult createFolder(String virtualFolderPath,
        String type, String newFolderName) throws Exception;
  protected abstract FileUploadResult fileUpload(String virtualFolderPath,
        String type, String filename, String contentType, java.io.File newFile)
        throws Exception;
  protected abstract void unknownCommand(String command, String virtualFolderPath,
        String type, String filename, String contentType, java.io.File newFile)
        throws Exception;
```

browse方法

本方法执行richtexteditor的'browse'命令指定的功能.

下表列出了实际的'browse'返回的结果名称.

Browse	Command Result Name
GetFolders	getFolders
GetFoldersAndFiles	getFoldersAndFiles
CreateFolder	createFolder
FileUpload	fileUpload

upload方法

本方法执行richtexteditor的'upload'命令指定的功能.
返回结果为'fileUpload'.

getFolders方法

当richtexteditor指定了'GetFolders'命令时将调用该方法. 该方法应当搜索服务器端并返回一个Folder[]数组.

要查询的文件夹路径由folderPath指定. 类型参数应该是下列三者之一: 'Image', 'Link'和'Flash'.

getFoldersAndFiles方法

当richtexteditor指定了'GetFoldersAndFiles'命令时将调用该方法. 该方法应当在virtualFolderPath指定的文件夹下搜索文件和文件夹并返回FoldersAndFiles对象.

要查询的文件夹路径由virtualFolderPath指定. 类型参数应该是下列三者之一: 'Image', 'Link'和'Flash'.

createFolder方法

当richtexteditor指定了'CreateFolder'命令时将调用该方法. 如果允许在服务器上创建文件夹, 该方法将创建一个新文件夹, 并使用一个CreateFolderResult对象将结果返回. CreateFolderResult类包括一个静态方法用于返回结果.

要查询的文件夹路径由virtualFolderPath指定. 类型参数应该是下列三者之一: 'Image', 'Link'和'Flash'.
.创建的新文件夹名称由newFolderName指定.

fileUpload方法

当richtexteditor指定了'FileUpload'命令时将调用该方法. 该方法将处理文件上传并返回FileUploadResult对象. FileUploadResult仅包含一个用于创建结果的静态方法.

要查询的文件夹路径由virtualFolderPath指定. 类型参数应该是下列三者之一: 'Image', 'Link'和'Flash'. 上传文件名称由filename指定, 文件的类型(content type)由conetnType指定, 文件的内容可以从newFile对象中读出.

结果类型

AbstractRichtexteditorResult

RichTextEditor所有结果的抽象基类. 它包括一些可以子类中使用的通用方法.

应该在xwork.xml(缺省就已经包含了)中加入如下的结果类型配置:

```
<!-- Results necessary when using 'browse server' and 'upload' feature of Richtexteditor -->
 <result-type name="richtexteditorGetFolders"
                class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorGetFoldersResult"
/>
 <result-type name="richtexteditorGetFoldersAndFiles"
                class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorGetFoldersAndFilesResult"
/>
 <result-type name="richtexteditorCreateFolder"
                class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorCreateFolderResult"
/>
 <result-type name="richtexteditorFileUpload"
                class="com.opensymphony.webwork.views.jsp.ui.RichtexteditorFileUploadResult"
/>
```

RichtexteditorGetFoldersResult

WebWork定义的结果, 创建符合要求的结果(使用XML格式)并写回到'GetFolder'命令对应的响应输出流中.

下面是一个响应结果的示例:

```
<?xml version="1.0" encoding="utf-8" ?>
```

```
<Connector command="GetFolders" resourceType="File">
  <CurrentFolder path="/Samples/Docs/" url="/UserFiles/File/Samples/Docs/" />
  <Folders>
    <Folder name="Documents" />
    <Folder name="Files" />
    <Folder name="Other Files" />
    <Folder name="Related" />
 </Folders>
</Connector>
```

RichtexteditorGetFoldersAndFilesResult

WebWork定义的结果, 创建符合要求的结果(使用XML格式)并写回到'GetFoldersAndFiles'命令对应的响应输出流中.

下面是一个响应结果的示例:

```
<?xml version="1.0" encoding="utf-8" ?>
<Connector command="GetFoldersAndFiles" resourceType="File">
  <CurrentFolder path="/Samples/Docs/" url="/UserFiles/File/Samples/Docs/" />
  <Folders>
    <Folder name="Documents" />
    <Folder name="Files" />
    <Folder name="Other Files" />
    <Folder name="Related" />
  </Folders>
  <Files>
    <File name="XML Definition.doc" size="14" />
    <File name="Samples.txt" size="5" />
    <File name="Definition.txt" size="125" />
    <File name="External Resources.drw" size="840" />
    <File name="Todo.txt" size="2" />
  </Files>
</Connector>
```

RichtexteditorCreateFolderResult

WebWork定义的结果, 创建符合要求的结果(使用XML格式)并写回到'CreateFolder'命令对应的响应输出流中.

下面是一个响应结果的示例:

```
<?xml version="1.0" encoding="utf-8" ?>
<Connector command="CreateFolder" resourceType="File">
  <CurrentFolder path="/Samples/Docs/" url="/UserFiles/File/Samples/Docs/" />
  <Error number="0" />
</Connector>
```

RichtexteditorUploadFileResult

WebWork定义的结果, 创建符合要求的结果(使用JavaScript的格式)并写回到'FileUpload'命令对应的响应输出流中.

下面是一个响应结果的示例:

```
<script type="text/javascript">
     window.parent.frames['frmUpload'].OnUploadCompleted(0) ;
</script>
```

select

This page last changed on May 24, 2006 by foxcoming.

⚠ 请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

创建一个HTML Select列表组件.

属性

名称	必填	缺省值	类型	描述
emptyOption	false	false	Boolean	是否在题头选项后面添加一个空的(--)选项
headerKey	false		Object/String	设置列表的题头主键值. 一定不能为空值! "'-1'"或"''"是正确的取值, ""是错误的取值.
headerValue	false		Object/String	列表的题头选项值
multiple	false		Object/String	创建一个多选列表. 如果value属性指定了一个数组(正确的元素类型), 那么将预先选中数组中指定的多个选项.
size	false		Integer	该组件列表框的大小(显示元素的个数)
list	**true**		Object/String	创建列表的可迭代数据源. 如果该列表是一个Map(key, value), 那么Map的主键将作为选项(<option>)的"value"属性, 而该主键对应的值作为选项的文本内容.
listKey	false		Object/String	列表数据源中元素对象的属性, 用于获取选项的值
listValue	false		Object/String	列表数据源中元素对象的属性, 用于获取选项的文本内容
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用

				缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress

onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
##: ######list###(select#######), ###OGNL#list########list#(######"months"),
####Map###(Key, months#####'01', '02'#)#####. '1'#####(Char), '01'######(String),
"1"######(String). ####, ###"value"####"list"##########(#: char#String###),
################. ######, ###########(auto-selected-entry)#.
```

```
<ww:select label="Pets"
       name="petIds"
       list="petDao.pets"
       listKey="id"
       listValue="name"
       multiple="true"
       size="3"
       required="true"
/>

<ww:select label="Months"
       name="months"
       headerKey="-1" headerValue="Select Month"
       list="#{'01':'Jan', '02':'Feb', [...]}"
       value="selectedMonth"
       required="true"
/>

// The month id (01, 02, ...) returned by the getSelectedMonth() call
// against the stack will be auto-selected
```

submit

This page last changed on May 25, 2006 by scud.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

绘制一个提交按钮. submit标签和form标签一起使用可以提供异步表单提交功能. submit可以绘制三种输出结果:

- input: 绘制<input type="submit"...>
- image: 绘制<input type="image"...>
- button: 绘制<button type="submit"...>

注意button类型已经演进为可以分别指定提交值和按钮上显示的文本, 但只能用于Microsoft Internet Explorer 6.0以上版本的浏览器.

下列属性只在使用AJAX时生效

- resultDivId
- notifyTopics
- onLoadJS
- preInvokeJS

远程表单(remote form)有三种使用模式, 分别使用resultDivId, notifyTopics或onLoadJS. 可以混合使用各种组合来达成期望的结果. 全部例子都包含在Ajax的样例Web项目中. Lets go through some scenarios to see how you might use it:

该标签可以工作在任何主题中, 但在ajax主题中与form标签组合使用尤为重要. 更多信息参见ajax theme.

属性

名称	必填	缺省值	类型	描述
resultDivId	false		String	包含提交结果的HTML元素的id(不能是表单或页面上任何其他元素的id).
onLoadJS	false		String	表单提交后将执行的Javascript代码. 格式为onLoadJS='yourMethodName(data,typ 注意: 如果你希望使用事件类型和返回数据的话, 必须保留参数data和type.
notifyTopics	false		String	表单提交完成后发送事件的主题 (Topic names to post an event to after the

				form has been submitted).
listenTopics	false		String	设置listenTopics属性.
preInvokeJS	false		String	在远程调用之前执行的Javascript代码. 格式为preInvokeJS='yourMethodName(data,
label	false		Object/String	提供提交按钮的显示文字(不同于提交值). 对于input类型的submit无效, 因为它的文本总是使用value参数的值. 对于image类型, alt参数将使用该值.
src	false		Object/String	为image类型提供图片地址. 对于input和button类型无效.
action	false		String	设置action属性.
method	false		String	设置method属性.
align	false		String	HTML的align属性.
type	false	input	String	submit的类型. 有效值为: input, button, image.
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在

				输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

例子1

```
<ww:submit value="%{'Submit'}" />
```

例子2

```
Render an image submit:
<ww:submit type="image" value="%{'Submit'}" label="Submit the form" src="submit.gif"/>
```

例子3

```
Render an button submit:
<ww:submit type="button" value="%{'Submit'}" label="Submit the form"/>
```

例子4

在另一个DIV中显示结果. 如果想在DIV中显示结果, 可以使用resultDivId指定用于显示结果的div的id. 这将使用InnerHtml方法插入结果内容. 结果将被加入到div中. 下面是这一方法的例子:

```
Remote form replacing another div:
<div id='two' style="border: 1px solid yellow;">Initial content</div>
<ww:form
      id='theForm2'
      cssStyle="border: 1px solid green;"
      action='/AjaxRemoteForm.action'
      method='post'
      theme="ajax">

  <input type='text' name='data' value='WebWork User' />
  <ww:submit value="GO2" theme="ajax" resultDivId="two" />

</ww:form >
```

例子5

通知其他控件(div)有变化发生. 可以采用发布-订阅(pub-sub)模型把控件变化消息通知其他控件以便执行相关的动作. 最常见的动作是执行某个action来刷新控件(这里的动作和控件是指订阅变化的控件在接到通知后执行的动作, 译注). notifyTopics可以实现这种功能. 可以使用逗号分隔多个主题名. 例如: notifyTopics="newPerson, dataChanged". 下面是这一方法的例子:

```
<ww:form id="frm1" action="newPersonWithXMLResult" theme="ajax"  >
    <ww:textfield label="Name" name="person.name" value="person.name" size="20" required="true"
/>
    <ww:submit id="submitBtn" value="Save" theme="ajax"  cssClass="primary"
notifyTopics="personUpdated, systemWorking" />
</ww:form >

<ww:div href="/listPeople.action" theme="ajax" errorText="error opps"
        loadingText="loading..." id="cart-body" >
    <ww:action namespace="" name="listPeople" executeResult="true" />
</ww:div>
```

例子6

传递JavaScript代码结果. 也就是返回结果中包含XML内容, 需要解析并用它完成一些很酷的效果. 这么做的方法是使用onLoadJS句柄(handler). 需要提供一个用于回调的包含结果(result)和事件类型(event type)参数的JavaScript函数名. 关键在于必须使用变量名'data'和'type'定义该回调函数. 例如: onLoadJS="myFancyDancyFunction(data, type)". 虽然本例中讨论的是XML, 但不仅限于XML, 回调函数中的data仅仅是你希望的返回结果. 下面是这一方法的例子:

```
<script language="JavaScript" type="text/javascript">
    function doGreatThings(data, type) {
        //Do whatever with your returned fragment...
        //Perhapps.... if xml...
              var xml = dojo.xml.domUtil.createDocumentFromText(data);
              var people = xml.getElementsByTagName("person");
              for(var i = 0;i < people.length; i ++){
                  var person = people[i];
                  var name = person.getAttribute("name")
                  var id = person.getAttribute("id")
                  alert('Thanks dude. Person: ' + name + ' saved great!!!');
              }

    }
</script>

<ww:form id="frm1" action="newPersonWithXMLResult" theme="ajax"  >
    <ww:textfield label="Name" name="person.name" value="person.name" size="20" required="true"
/>
    <ww:submit id="submitBtn" value="Save" theme="ajax"  cssClass="primary"
onLoadJS="doGreatThings(data, type)" />
</ww:form>
```

textarea

This page last changed on May 20, 2006 by scud.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

提供了一个HTML textarea标签

属性

名称	必填	缺省值	类型	描述
cols	false		Integer	HTML cols 属性
readonly	false	false	Boolean	设置为只读,不允许输入
rows	false		Integer	HTML rows 属性
wrap	false		String	HTML wrap 属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(

				译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:textarea label="Comments" name="comments" cols="30" rows="8"/>
```

textfield

This page last changed on May 05, 2006 by dogg.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

绘制一个text类型的HTML input元素.

属性

名称	必填	缺省值	类型	描述
maxlength	false		Integer	HTML maxlength属性
maxLength	false		Object/String	不建议使用. 建议使用maxlength属性替代.
readonly	false	false	Boolean	设置为只读,不允许输入
size	false		Integer	HTML size 属性
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在

				输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

在这个例子中显示了一个text表单控件,它的label属性将从ResourceBundle调用ActionSupport的 getText()检索资源.

```
<ww:textfield label="%{text('user_name')}" name="user" />
```

token

This page last changed on May 21, 2006 by dogg.

请确认你已经读过了Tag Syntax文档,并且理解标签的属性语法是如何工作的.

描述

防止多次提交表单.
使用token标签能帮助解决多次提交表单的问题.此标签需要你启用TokenInterceptor 或者TokenSessionInterceptor拦截器.
ww:token标签只不过放置了一个隐藏的表单元素,它包含一个唯一的令牌.

属性

名称	必填	缺省值	类型	描述
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的

				value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:token />
```

updownselect

This page last changed on May 24, 2006 by foxcoming.

请确认你已经读过了 Tag Syntax 文档 并且理解标签的属性语法是如何工作的.

描述

创建一个元素Select列表组件, 带有可以上下移动选项元素的按钮. 当表单提交时, 列表中的元素全部被选中并按照排列顺序(从顶至底)被提交.

属性

名称	必填	缺省值	类型	描述
allowMoveUp	false	true	Boolean	是否显示"上移"按钮
allowMoveDown	false	true	Boolean	是否显示"下移"按钮
allowSelectAll	false	true	Boolean	是否显示"全选"按钮
moveUpLabel	false	ˆ	String	"上移"按钮的文本
moveDownLabel	false	v	String	"下移"按钮的文本
selectAllLabel	false	*	String	"全选"按钮的文本
emptyOption	false	false	Boolean	是否在题头选项后面添加一个空的(--)选项
headerKey	false		Object/String	设置列表的题头主键值. 一定不能为空值! "'-1'"或"''"是正确的取值, ""是错误的取值.
headerValue	false		Object/String	列表的题头选项值
multiple	false		Object/String	创建一个多选列表. 如果value属性指定了一个数组(正确的元素类型), 那么将预先选中数组中指定的多个选项.
size	false		Integer	该组件列表框的大小(显示元素的个数)
list	**true**		Object/String	创建列表的可迭代数据源. 如果该列表是一个Map(key, value), 那么Map的主键将作为选项(<option>)的"value"属性, 而该主键对应的值作为选项的文本内容.
listKey	false		Object/String	列表数据源中元素对象的属性, 用于获取

				选项的值
listValue	false		Object/String	列表数据源中元素对象的属性，用于获取选项的文本内容
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:就是html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true，在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove

onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<!-- Example 1: simple example -->
<ww:updownselect
list="#{'england':'England', 'america':'America', 'germany':'Germany'}"
name="prioritisedFavouriteCountries"
headerKey="-1"
headerValue="--- Please Order Them Accordingly ---"
emptyOption="true" />

<!-- Example 2: more complex example -->
<ww:updownselect
list="defaultFavouriteCartoonCharacters"
name="prioritisedFavouriteCartoonCharacters"
headerKey="-1"
headerValue="--- Please Order ---"
emptyOption="true"
allowMoveUp="true"
allowMoveDown="true"
allowSelectAll="true"
moveUpLabel="Move Up"
moveDownLabel="Move Down"
selectAllLabel="Select All" />
```

FreeMarker Tags

This page last changed on Apr 19, 2006 by tin.

FreeMarker标签是WebWork提供的一般 标签 的扩展. 只要简单的了解这些标签的访问方式: <@ww.xxx> ...</@ww.xxx> (这里的xxx指WebWork支持的标签), 就可以马上开始使用了.

语法

例如, 在JSP中你可能这样创建一个form:

```
<ww:form action="updatePerson">
    <ww:textfield label="First name" name="firstName"/>
    <ww:submit value="Update"/>
</ww:form>
```

FreeMarker中创建同样的form是这样的:

```
<@ww.form action="updatePerson">
    <@ww.textfield label="First name" name="firstName"/>
    <@ww.submit value="Update"/>
</@ww.form>
```

这就基本上覆盖了使用FreeMarker标签的所有内容了, 还有一些高级功能你应该了解, 特别是属性和参数如何共同工作, 还有参数的类型 (String, List, 等等) 能够影响到标签的行为.

属性和参数

与老版本的JSP不同 (JSP标签 的基础), FreeMarker 允许 动态属性, 很就像JSP 2.0. 这就意味着你可以给这些标签提供这些标签本不支持的属性. 这些不能够直接被应用到标签对象的属性将会被放到这个标签的通用 `parameters` map.

例如, 假设你的JSP中有如下代码:

```
<ww:url value="somePage">
    <ww:param name="personId" value="%{personId}"/>
</ww:url>
```

在FreeMarker中, 你可以将它简化为:

```
<@ww.url value="somePage" personId="${personId}"/>
```

不仅可以替代原本要使用 param 的情况, 你还可以在扩展你的 Form标签 的模版或者主题的时候使用这个功能. 例如, 假设你创建了一个 "三分栏" 主题来替代标准的两分栏主题 (xhtml). 你可能想要在第三栏中显示一个叫做"description"的扩展参数. 你的form可以这样写:

```
<@ww.form action="updatePerson">
    <@ww.textfield label="First name" name="firstName" description="..."/>
    <@ww.submit value="Update"/>
```

```
</@ww.form>
```

And then in your new template you can refer to the description using ${parameters.description}.
然后在你的新模版中你可以通过 ${parameters.description} 应用这个description.

> 有时你也许还是希望使用 param 标签, 例如当你需要在标签中嵌入复杂的HTML时. param标签能够提供FreeMarker所提供的内联(inline)属性以外的属性: 它可以将param标签内的所有内容作为 _value_attribute 提供访问.

属性类型

记住所有的标签属性都首先设定为String类型 - 然后它们会被解析 (使用 OGNL) 为不同类型, 例如List, int, 或者boolean. 一般情况下这样就够了, 但是在使用FreeMarker的时候它可能会成为一种限制, FreeMarker提供了更先进的方式来设定属性. 假设如下的例子:

```
<@ww.select label="Foo label - ${foo}" name="${name}" list="%{{1, 2, 3}}"/>
```

这里会发生的是每个属性将会被最佳匹配并解析为string. 这可能会调用hash的内部FreeMarker对象的toString()方法, 所有的对象最后将会与你所预期的一样. 这样, 当标签运行时, _list_attribute 将会被从String转化为List, 通过 OGNL 的先进集合支持.

但是假设你希望使用FreeMarker的list后者hash支持来替代呢? 你可以这样做:

```
<@ww.select label="Foo label - ${foo}" name="${name}" list={1, 2, 3}/>
```

注意这个list属性两边已经没有了引号. 现在它将会作为一个对象进入标签, 它将不容易被转换为String. 一般的, 标签将会调用toString(), 它将返回"[1, 2, 3]", 而这个字符串将不能被OGNL转换为List. 比来回传递要好, WebWork里面的FreeMarker标签支持将会识别出这个集合并且不会讲他们传送到一般的标签属性中, 而是讲他们直接注入到 parameters map , 等待模版来访问.

最后, 所有都会像你期望的那样运转, 但是它能够帮助你理解OGNL什么时候会被使用而什么时候不会, 还有属性类型是如何被转换的.

JSP标签支持

虽然WebWork提供了原生的FreeMarker标签, 但是你可能希望使用那些只能在JSP中使用的第三方标签. 幸运的是, FreeMarker 能够运行JSP标签. 如果希望这样做, 你必须配置 web.xml 2.1.x 兼容性 中描述的 JspSupportServlet, 这样就允许FreeMarker能够访问所需要的对象来模拟一个JSP标签库运行容器.

当你那样配置好后, 你可以类似这样的在你的模版中加入:

```
<#assign cewolf=JspTaglibs["/WEB-INF/cewolf.tld"] />
...
<@cewold.xxx ... />
```

JSP Tags

This page last changed on Apr 19, 2006 by tin.

JSP标签是WebWork提供的一般Tag的扩展. 只要简单的了解这些标签的访问方式:<ww:xxx> ... </ww:xxx>(这里的xxx指WebWork支持的标签), 就可以马上开始使用了.

标签库(Tag Library)定义(TLD)

JSP TLD包含于 **webwork.jar** 中, 你可以按照 web.xml 文档中解释的那样定义TLD. 如果你做好了, 你可以在你的JSP中添加如下声明:

```
<%@ taglib prefix="ww" uri="webwork" %>
```

现在你可以使用标签了, 就像这样:

```
<ww:iterator value="people">
    <ww:property value="lastName"/>, <ww:property value="firstName"/>
</ww:iterator>
```

Non Form Tags

This page last changed on May 13, 2006 by scud.

1. a - 输出一个锚连接(link)
2. actionerror - 如果存在的话输出Action错误消息
3. actionmessage - 如果存在的话输出Action消息
4. component - 在使用此标签的地方通过主题(theme)、模板(template)属性输出一个非常个性化的组件.
5. date - 输出一个日期
6. div - 输出一个div片断
7. fielderror - 如果存在的话输出字段的出错信息
8. panel - 输出选项卡
9. table - 输出表格
10. tabbedpane - (不赞成使用,建议使用tabbedPanel)
11. tabbedPanel - 输出选项卡的一个Panel
12. tree - 输出一个树型组件
13. treenode - 在树型组件内绘制树的节点

a

This page last changed on May 20, 2006 by scud.

描述

A标签创建一个HTML 代码 <a href='' />，当被点击时，会通过远程XMLHttpRequest调用访问一个网址，这是通过dojo框架完成的．从URL返回的结果被当作JavaScript处理．如果提供了一个 "listenTopics"参数，当结果返回时，它会发布一个"click"消息给那些监听的topic.

虽然这个标签可以在simple theme， xhtml theme和其他theme中使用，它实际上是为了更好地和ajax theme工作而设计的．我们推荐阅读ajax a template文档了解更多信息.

参数

名称	必填	缺省值	类型	描述
id	false		String	组件的id
notifyTopics	false		Object/String	远端调用发生后发送信息的Topic 名字
preInvokeJS	false		String	一段在执行目标网址之前需要调用的javascript片断． 如果设置了则必须返回true或false． True标识继续执行，false标识不需要继续执行．可能用来显示确认对话框.
theme	false		Object/String	输出元素时使用的主题（theme） **这个标签通常使用ajax theme**
href	false		String	用来调用获取内容的URL
errorText	false		String	如果获取内容时发生错误显示给用户的文字
showErrorTransportText	false	false	Boolean	当访问URL有问题时是否把错误信息当作内容显示
afterLoading	false		String	获取内容完成后要执行的Javascript代码
openTemplate	false		Object/String	用来输出开始的html的模板.
template	false		Object/String	输出元素时使用的模板（template）（不使用缺省的）
cssClass	false		Object/String	输出元素时的class属性

cssStyle	false		Object/String	输出元素时的css样式定义
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置

				html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置

使用

为了启用这个功能,使用head标签和ajax theme.浏览ajax head template了解更多信息.然后在ajax a template里浏览具体使用情况.

如果你想要在你的ww:a里使用额外的参数,最好的办法是使用一个ww:url来创建你的url,然后传递这个url到你的ww:a标签.这通过创建一个ww:url并指定一个id属性,就像这里例子里面的 "testUrlId".然后ww:a标签在href属性里通过 "%{testUrlId}" 引用这个id.

```
<ww:url id="testUrlId" namespace="/subscriber" action="customField" method="delete">
                  <ww:param name="customFieldDefinition.id" value="${id}"/>
              </ww:url>
             <ww:a errorText="Sorry your request had an error." preInvokeJS="confirm('Are you
sure you want to delete this item?')" href="%{testUrlId}">
                    <img src="<ww:url value="/images/delete.gif"/>" border="none"/></ww:a>
```

actionerror

This page last changed on May 11, 2006 by dogg.

描述

如果存在的话输出Action错误消息,它的输出布局依赖于使用的主题(theme).

属性

名称	必填	缺省值	类型	描述
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置

				html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:actionerror />
   <ww:form .... >>
      ....
   </ww:form>
```

actionmessage

This page last changed on May 11, 2006 by dogg.

描述

如果存在的话输出Action消息,它的输出布局依赖于使用的主题(theme).

属性

名称	必填	缺省值	类型	描述
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置

				html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:actionmessage />
<ww:form .... >
   ....
</ww:form>
```

component

This page last changed on May 18, 2006 by scud.

描述

使用特定的模板输出一个自定义的UI widget(组件).附加的对象可以通过param标签传递给模板.设置的对象可以在模板里面通过 $parameters.paramname 获取.

在后面的JSP和Velocity例子里,两个参数被传递给组件.在component(组件)内部,它们可以通过 $parameters.get('key1') 和 $parameters.get('key2') 的方式被访问.Velocity也允许你通过 $parameters.key1 和 $parameters.key2来引用它们.

目前,你的自定义UI组件可以使用Velocity,JSP或者Freemarker编写,通过文件的后缀会找到正确的处理引擎来处理.

注意: value参数总是会被在OgnlValueStack上进行运算,因此如果你想传递一个字符串常量给你的组件,确保它包含在引号中,例如 value="'value1'",否则,value stack会在stack上搜索一个包含方法名为getValue1()的对象.(我是这么写的,但是我不完全确定是这样的.我会很快验证它)

参数

名称	必填	缺省值	类型	描述
theme	false		Object/String	用来输出元素的theme(不使用缺省)
template	false		Object/String	用来输出元素的template(不使用缺省)
cssClass	false		Object/String	元素使用的css class
cssStyle	false		Object/String	元素使用的css style
title	false		Object/String	设置输出的html元素的html title属性
disabled	false		Object/String	设置输出的html元素的属性 disabled
label	false		Object/String	输出一个元素的label的表达式
labelPosition	false	left	Object/String	不推荐.
labelposition	false		Object/String	定义表单元素的label的位置 (top/left)
requiredposition	false		Object/String	定义必填元素的位置 (left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true,输出的元素会被标识为必填
tabindex	false		Object/String	设置输出的html元素的tabindex属性
value	false		Object/String	预设input元素的

				value.
onclick	false		Object/String	设置html元素的onclick属性
ondblclick	false		Object/String	设置html元素的ondblclick
onmousedown	false		Object/String	设置html元素的onmousedown 属性
onmouseup	false		Object/String	设置html元素的onmouseup 属性
onmouseover	false		Object/String	设置html元素的onmouseover属性
onmousemove	false		Object/String	设置html元素的onmousemove属性
onmouseout	false		Object/String	设置html元素的onmouseout属性
onfocus	false		Object/String	设置html元素的onfocus 属性
onblur	false		Object/String	设置html元素的onblur 属性
onkeypress	false		Object/String	设置html元素的onkeypress属性
onkeydown	false		Object/String	设置html元素的onkeydown属性
onkeyup	false		Object/String	设置html元素的onkeyup属性
onselect	false		Object/String	设置html元素的onselect属性
onchange	false		Object/String	设置html元素的onchange属性
tooltip	false		String	设置特别组件的tooltip属性
tooltipConfig	false		String	设置tooltip元素的配置
id	false		Object/String	元素的id. 对于UI和表单标签它会被用作HTML的id属性

例子

JSP

```
<ww:component template="/my/custom/component.vm"/>
```

或者

```
<ww:component template="/my/custom/component.vm">
      <ww:param name="key1" value="value1"/>
      <ww:param name="key2" value="value2"/>
    </ww:component>
```

Velocity

```
#tag( Component "template=/my/custom/component.vm" )
```

或者

```
#bodytag( Component "template=/my/custom/component.vm" )
      #param( "key1" "value1" )
      #param( "key2" "value2" )
    #end
```

date

This page last changed on May 18, 2006 by scud.

描述

用不同的方式格式化输出Date对象.

date标签能帮助你非常容易和快速的格式化一个日期对象.你能自定义格式(例如."dd/MM/yyyy hh:mm"),产生非常容易识别和读懂的输出(比如像"in 2 hours, 14 minutes"),你能在你的属性配置文件中添加'webwork.date.format' 键来预定义格式.

如果你的属性配置文件中没有定义此键,它最后输出的将是缺省的DateFormat.MEDIUM格式.

注意:如果被请求的Date对象在堆栈中没有找到,返回的将是空白.

可配置的属性如下:

- name
- nice
- format

下列date组件的工作方式依赖于nice属性(缺省的值是false)和format属性的设置值.

情况1:当nice属性设为true时

i18n key	缺省值
webwork.date.format.past	{0} ago
webwork.date.format.future	in {0}
webwork.date.format.seconds	an instant
webwork.date.format.minutes	{0,choice,1#one minute|1<{0} minutes}
webwork.date.format.hours	{0,choice,1#one hour|1<{0} hours}{1,choice,0#|1#, one minute|1<, {1} minutes}
webwork.date.format.days	{0,choice,1#one day|1<{0} days}{1,choice,0#|1#, one hour|1<, {1} hours}
webwork.date.format.years	{0,choice,1#one year|1<{0} years}{1,choice,0#|1#, one day|1<, {1} days}

情况2:当nice属性设为false并且设置了format属性时(例如.dd/MM/yyyyy)

在此情况下将使用format属性设置的格式.

情况3:当nice属性设为false并且没有指定format属性时

i18n key	缺省值
webwork.date.format	如果也没有找到此键将使用DateFormat.MEDIUM格式

属性

名称	必填	缺省值	类型	描述
format	false		Object/String	Date或DateTime的格式化模板
nice	false	false	Boolean	是否输出日期的更细化表达
name	true		String	要格式化的date值
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<ww:date name="person.birthday" format="dd/MM/yyyy" />
<ww:date name="person.birthday" format="%{getText('some.i18n.key')}" />
<ww:date name="person.birthday" nice="true" />
<ww:date name="person.birthday" />
```

div

This page last changed on May 20, 2006 by scud.

描述

此div标签是一个AJAX标签,提供了从当前页面发起一个远程调用,可以在不刷新整个页面的情况下更新部分内容.
它创建一个HTML的<DIV/>标签,通过dojo框架的远程XMLHttpRequest获取内容.
如果设置了"listenTopics"属性,它会监听那些topic,当收到任何接收到的信息它会刷新自己的内容.

此标签可以在 simple主题, xhtml主题, 和其他主题中使用, 但此标签的最好工作主题是 ajax主题. 我们推荐你阅读 ajax div template 文档获取更多信息.

参数

名称	必填	缺省值	类型	描述
updateFreq	false	0	Integer	重新获取内容的间隔时间 (毫秒)
delay	false	0	Integer	开始获取内容前的等待时间(毫秒)
loadingText	false		Object/String	当获取新内容时显示给用户的文字 (如果获取内容需要花费很长时间这就特别好)
listenTopics	false		Object/String	要监听的Topic名字(逗号分割),这会导致重新获取DIV的内容
theme	false		Object/String	输出元素时使用的主题(theme) **这个标签通常使用ajax theme**
href	false		String	用来调用获取内容的URL
errorText	false		String	如果获取内容时发生错误显示给用户的文字
showErrorTransportText	false	false	Boolean	当访问URL有问题时是否把错误信息当作内容显示
afterLoading	false		String	获取内容完成后要执行的Javascript代码
openTemplate	false		Object/String	用来输出开始的html的模板.
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性

cssStyle	false		Object/String	输出元素时的css样式定义
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置

				html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

用法

为了启用这个标签, 需要使用 head 标签和 ajax theme. 查看 ajax head template 获取更多信息. 要了解更多使用方法的详细信息请查看 ajax div template.

✓ **提示**

我发现这个div的target必须返回结构良好的html.主要是一个html和一个body块必须正常地关闭.

fielderror

This page last changed on May 11, 2006 by scud.

描述

如果存在的话输出字段的出错信息,它的输出布局依赖于使用的主题(theme).

属性

名称	必填	缺省值	类型	描述
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置

				html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<!-- ## 1 -->
   <ww:fielderror />

   <!-- ## 2 -->
   <ww:fielderror>
        <ww:param>field1</ww:param>
        <ww:param>field2</ww:param>
   </ww:fielderror>
   <ww:form .... >>
      ....
   </ww:form>

   OR

   <ww:fielderror>
                <ww:param value="%{'field1'}" />
                <ww:param value="%{'field2'}" />
   </ww:fielderror>
   <ww:form .... >>
      ....
   </ww:form>
```

例子1:显示所有字段出错信息.
例子2:仅仅显示'field1'和'field2'字段的出错信息.

panel

This page last changed on May 20, 2006 by scud.

描述

输出一个tabbedPanel的面板.

参数

名称	必填	缺省值	类型	描述
tabName	**true**		Object/String	显示在选项卡列表头部的文字
subscribeTopicName	false		Object/String	设置subscribeTopicName属性
remote	false	false	Boolean	确定这是否是一个远程(remote)panel(ajax)或者一个本地panel (内容装载到一个可视/隐藏的容器)
updateFreq	false	0	Integer	重新获取内容的间隔时间 (毫秒)
delay	false	0	Integer	开始获取内容前的等待时间(毫秒)
loadingText	false		Object/String	当获取新内容时显示给用户的文字 (如果获取内容需要花费很长时间这就特别好)
listenTopics	false		Object/String	要监听的Topic名字(逗号分割),这会导致重新获取DIV的内容
theme	false		Object/String	输出元素时使用的主题(theme) **这个标签通常使用ajax theme**
href	false		String	用来调用获取内容的URL
errorText	false		String	如果获取内容时发生错误显示给用户的文字
showErrorTransportText	false	false	Boolean	当访问URL有问题时是否把错误信息当作内容显示
afterLoading	false		String	获取内容完成后要执行的Javascript代码
openTemplate	false		Object/String	用来输出开始的html的模板.
template	false		Object/String	输出元素时使用的模

				板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置

				html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

下面是一个tabbedpanel的例子,panel标签利用了local和remote内容.
如果你寻找一个"漂亮的"圆角外观,有一个额外的配置.它假设tab的背景是白色的.如果你用了一个不同的颜色,请修改Rounded()方法的参数.

```
<link rel="stylesheet" type="text/css" href="<ww:url value="/webwork/tabs.css"/>">
<link rel="stylesheet" type="text/css" href="<ww:url
value="/webwork/niftycorners/niftyCorners.css"/>">
<link rel="stylesheet" type="text/css" href="<ww:url
value="/webwork/niftycorners/niftyPrint.css"/>" media="print">
<script type="text/javascript" src="<ww:url value="/webwork/niftycorners/nifty.js"/>"></script>
<script type="text/javascript">
    dojo.event.connect(window, "onload", function() {
        if (!NiftyCheck())
            return;
        Rounded("li.tab_selected", "top", "white", "transparent", "border #ffffffS");
        Rounded("li.tab_unselected", "top", "white", "transparent", "border #ffffffS");
        // "white" needs to be replaced with the background color
    });
</script>
```

tabbedpane

This page last changed on May 11, 2006 by dogg.

不赞成使用

此标签不赞成使用,已经由TabbedPanel替代.

tabbedpane

名称	类型	必填	缺省值	描述
id	string	FALSE		
contentName	string	TRUE		name of collection to use
selectedIndex	integer	FALSE		
name	string	TRUE		
value	string	FALSE		
required	boolean	FALSE		
disabled	boolean	FALSE		
theme	string	FALSE		
template	string	FALSE	tabbedpane	
cssClass	string	FALSE		
cssStyle	string	FALSE		
label	string	FALSE		
labelposition	string	FALSE		
tabindex	string	FALSE		
onclick	string	FALSE		
ondblclick	string	FALSE		
onmousedown	string	FALSE		
onmouseup	string	FALSE		
onmouseover	string	FALSE		
onmousemove	string	FALSE		
onmouseout	string	FALSE		
onfocus	string	FALSE		
onblur	string	FALSE		
onkeypress	string	FALSE		
onkeydown	string	FALSE		
onselect	string	FALSE		
onchange	string	FALSE		

tabbedPanel

This page last changed on May 20, 2006 by scud.

描述

tabbedpanel部件是一个AJAX组件,每个选项卡可以是本地的内容或者远程的内容(每次用户选择这个选项时刷新).

这个标签仅在ajax theme中工作.在使用这个标签之前请确认研读这个theme.

参数

名称	必填	缺省值	类型	描述
id	**true**		String	设置给组件的id.
openTemplate	false		Object/String	用来输出开始的html的模板.
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex

value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置

例子

下面是一个tabbedpanel和panel的例子,利用了本地(local)和远程(remote)内容.

```
<ww:tabbedPanel id="test2" theme="simple" >
      <ww:panel id="left" tabName="left" theme="ajax">
          This is the left pane<br/>
          <ww:form >
              <ww:textfield name="tt" label="Test Text" />  <br/>
              <ww:textfield name="tt2" label="Test Text2" />
          </ww:form>
      </ww:panel>
      <ww:panel remote="true" href="/AjaxTest.action" id="ryh1" theme="ajax" tabName="remote
one" />
      <ww:panel id="middle" tabName="middle" theme="ajax">
```

```
            middle tab<br/>
            <ww:form >
                <ww:textfield name="tt" label="Test Text44" />  <br/>
                <ww:textfield name="tt2" label="Test Text442" />
            </ww:form>
        </ww:panel>
        <ww:panel remote="true" href="/AjaxTest.action"  id="ryh21" theme="ajax" tabName="remote
right" />
  </ww:tabbedPanel>
```

如果你寻找一个"漂亮的"圆角外观,有一个额外的配置.它假设tab的背景是白色的.如果你用了一个不同的颜色,请修改Rounded()方法的参数.

```
<link rel="stylesheet" type="text/css" href="<ww:url value="/webwork/tabs.css"/>">
<link rel="stylesheet" type="text/css" href="<ww:url
value="/webwork/niftycorners/niftyCorners.css"/>">
<link rel="stylesheet" type="text/css" href="<ww:url
value="/webwork/niftycorners/niftyPrint.css"/>" media="print">
<script type="text/javascript" src="<ww:url value="/webwork/niftycorners/nifty.js"/>"></script>
<script type="text/javascript">
    dojo.event.connect(window, "onload", function() {
        if (!NiftyCheck())
            return;
        Rounded("li.tab_selected", "top", "white", "transparent", "border #ffffffS");
        Rounded("li.tab_unselected", "top", "white", "transparent", "border #ffffffS");
        // "white" needs to be replaced with the background color
    });
</script>
```

table

This page last changed on May 18, 2006 by scud.

属性

名称	必填	缺省值	类型	描述
modelName	true		String	model使用的名字
sortColumn	false		Integer	需要排序的列
sortOrder	false	NONE	String	设置排序的顺序. 允许的值为 NONE, ASC 和 DESC
sortable	false	false	Boolean	是否表格为可以排序的. 如果设置为true,model必须实现接口 com.opensymphony.webwork.componen
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的

				value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

tree

This page last changed on May 18, 2006 by scud.

描述

输出一个支持AJAX的树型组件.

属性

名称	必填	缺省值	类型	描述
toggle	false		Object/String	toggle 属性.
treeSelectedTopic	false		Object/String	treeSelectedTopic 属性.
treeExpandedTopic	false		Object/String	treeExpandedTopic 属性.
treeCollapsedTopic	false		Object/String	treeCollapsedTopic 属性.
openAll	false	false	boolean	openAll 属性.
rootNode	false		Object/String	rootNode 属性.
childCollectionProperty	false		Object/String	childCollectionProperty 属性.
nodeTitleProperty	false		Object/String	nodeTitleProperty 属性.
nodeIdProperty	false		Object/String	nodeIdProperty 属性 .
openTemplate	false		Object/String	设置输出Html的开始部分(opening)的模板 .
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式

labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex
value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置

id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<tree .../>
```

treenode

This page last changed on May 18, 2006 by scud.

描述

输出一个支持AJAX树型组件的节点.

属性

名称	必填	缺省值	类型	描述
label	**true**		Object/String	用来输出树型节点label的表达式.
openTemplate	false		Object/String	设置用来输出html的开始部分(opening)的模板.
theme	false		Object/String	输出元素时使用的主题(theme)(不使用缺省的)
template	false		Object/String	输出元素时使用的模板(template)(不使用缺省的)
cssClass	false		Object/String	输出元素时的class属性
cssStyle	false		Object/String	输出元素时的css样式定义(译者注:html元素的style属性)
title	false		Object/String	在输出元素时设置html属性title
disabled	false		Object/String	在输出元素时设置html属性disabled
label	false		Object/String	用于输出一个元素对应的label的表达式
labelPosition	false	left	Object/String	不赞成使用.
labelposition	false		Object/String	定义元素标签的位置(top/left)
requiredposition	false		Object/String	定义required属性输出的位置(left|right)
name	false		Object/String	元素的名字
required	false	false	Boolean	如果设置为true, 在输出标签时将显示出此字段是必须输入的(译者注:如果使用默认模板,将会标示为"*")
tabindex	false		Object/String	在输出元素时设置html属性tabindex

value	false		Object/String	预设input元素的value属性.
onclick	false		Object/String	在输出元素时设置html属性onclick
ondblclick	false		Object/String	在输出元素时设置html属性ondblclick
onmousedown	false		Object/String	在输出元素时设置html属性onmousedown
onmouseup	false		Object/String	在输出元素时设置html属性onmouseup
onmouseover	false		Object/String	在输出元素时设置html属性onmouseover
onmousemove	false		Object/String	在输出元素时设置html属性onmousemove
onmouseout	false		Object/String	在输出元素时设置html属性onmouseout
onfocus	false		Object/String	在输出元素时设置html属性onfocus
onblur	false		Object/String	在输出元素时设置html属性onblur
onkeypress	false		Object/String	在输出元素时设置html属性onkeypress
onkeydown	false		Object/String	在输出元素时设置html属性onkeydown
onkeyup	false		Object/String	在输出元素时设置html属性onkeyup
onselect	false		Object/String	在输出元素时设置html属性onselect
onchange	false		Object/String	在输出元素时设置html属性onchange
tooltip	false		String	设置元素的tooltip属性(译者注:tooltip为工具栏提示)
tooltipConfig	false		String	设置tooltip属性的配置
id	false		Object/String	id是定位元素时使用的. 对于UI和表单标签它会被用作HTML的id属性

例子

```
<treenode .../>
```

Tag Syntax

This page last changed on Apr 30, 2006 by scud.

WebWork的标签语法非常容易理解,为了快速开始,所有你需要知道的就是所有的属性最初都是被设置为字符串的.然后它们会解析 %{ ... } 语法,任何大括号之间的内容都会基于value stack求值.

升级事项!
标签语法不总是这么简单 - 如果你从 WebWork 2.1.7 或者前面的版本升级过来的,你可能希望阅读一下 Alt Syntax.

就像生活里的其他事情,它原来并不是十分 那么 简单.明确地说,实际上有3个规范你要知道:

1. 所有 String 属性类型的都会 解析 %{ ... } 中间的字符.
2. 所有 非字符串 属性类型都 不 会解析,但是会直接被当作一个OGNL表达式求值.
3. 对第二个规则的例外情况是如果 非字符串 属性以 %{ 开始并以 } 结束, 这些字符在对表达式求值之前会被截取出来.

理解这些规则的最好方法就是查看一些例子.

我们成人这些规范可能令人糊涂.通常情况下,你根本不需要知道它们,在99.9%的时间里一切事情都会"正常工作".当然,就像我们在例子里看到的,会有一些棘手的情况需要理解这些规范.在WebWork将来的版本中,会努力让标签语法更简单

一些例子

最简单的例子来解释标签语法如何工作就是下面的例子.这个例子仅仅演示了规则1:

```
<ww:textfield label="%{getText("state.label")}" name="state"/>
```

在这个例子里,label被动态求值,设置为OGNL表达式 getText("state.label") 的结果,这个表达式会调用国际化装置来获取i18n主键 state.label 的值.name,是一个字符串属性,简单地被设置为 state 字符串.

下一个例子演示了规则2:

```
<ww:select label="%{getText("state.label")}" name="state" multiple="true"/>
```

这个例子看起来类似前一个例子,要注意到的主要事情是 multiple 属性是 Boolean 类型的, 这意味这它符合规则2.通常你不会注意到这个,因为 "true" 会被当作OGNL表达式求值得到true,也就是你想要的.

现在让我们假设我们要扩展这个例子来演示规则3,让multiple属性变成动态的:

```
<ww:select label="%{getText("state.label")}" name="state" multiple="%{allowMultiple}"/>
```

因为这个属性是 Boolean 类型的,并且以规则3中正确的字符开始和结束,它变成表达式 allowMultiple, 并会在value stack上进行求值,返回一个true或者false,就像前一个例子一样.

无论如何要关注有一个棘手的例子.例如,下面的多半是 **不正确的**:

```
<ww:textfield label="%{getText("state.label")}" name="state" value="CA"/>
```

这个例子只有在如果表达式 CA 会产生某些东西时才能工作,也就是说你的action有一个 getCA() 方法,这大概不是你所期望的.这是因为 value 属性是 Object ,因此规则2会被应用.如果期望设置一个静态的字符串作为初始值,你会需要提供一个OGNL表达式来返回一个字符串.例如,这是来完成它的 **正确** 方式:

```
<ww:textfield label="%{getText("state.label")}" name="state" value="%{'CA'}"/>
```

同时你也可以设置value属性仅仅为 "'CA'",我们推荐使用被解析表达式的方式,因为将来当WebWork支持解析所有类型的属性时,你的代码会继续工作.

Alt Syntax

This page last changed on Apr 12, 2006 by scud.

altSyntax 是一个可以在webwork.properties里定义的选项. 缺省它被设置为true,而且 **强烈** 推荐你不要改变它,除非你是从WebWork 2.1.7或者更前面的版本升级过来的.

迁移提示
你可以通过set标签基于每个页面开启 altSyntax. 简单地设置name为_useAltSyntax_ 变量的value 为 true. 此时,所有的标签都会使用 altSyntax 来处理剩余的部分请求.

altSyntax改变了标签如何被解析的行为.它不再在value stack上对每个标签参数进行求值,也不需要用单引号包含字符串文字,它仅对标记的表达式进行求值.

举例:

下面的代码使用了Tag Syntax:

```
<ww:iterator value="cart.items">
   ...
   <ww:textfield label="'Cart item No.' + #rowstatus.index + ' note'"
                 name="'cart.items[' + #rowstatus.index + '].note'"
                 value="note" />
</ww:iterator>
```

这感觉有点不像HTML标签的行为,而且你多写了单引号.现在下面是同样例子使用了altSyntax:

```
<ww:iterator value="cart.items">
   ...
   <ww:textfield label="Cart item No. %{#rowstatus.index} note"
                 name="cart.items[%{#rowstatus.index}].note"
                 value="%{note}" />
</ww:iterator>
```

只有包含在%{}里面的表达式才会被求值.代码短小而且清晰,非常类似JSTL EL的用法.引用问题,例如对javascript函数的调用,就被避免了.

为了全面理解为什么这个选项存在以及差别是什么,最好了解一点WebWork的历史.

如果你 不是 从WebWork 2.1.7或者更前面的版本升级过来的,你无须关心WebWork发展的历史,你可以略过此部分.
查看 Tag Syntax 部分来了解更多关于标准标签语法支持的信息

历史

在WebWork 2.1.4中,altSyntax选项被引进.WebWork in Action这本书,它是基于WebWork 2.1.7的,是完全基于altSyntx语法被开启的前提下编写的.在WebWork 2.2中,altSyntax缺省被开启,而且最终老的语法不在被支持而且会从代码里移除.

Themes and Templates

This page last changed on Apr 29, 2006 by scud.

WebWork提供的Html 标签的核心是 **themes** 和 **templates**.我们首先定义三个关键术语:

- **tag** - 一小段代码,在 JSP, FreeMarker 或者 Velocity 里执行.
- **template**(模板) - 一点代码,通常使用 FreeMarker 编写, 能被特定的标签(HTML tags)输出
- **theme** - 一系列 templates 打包在一起,提供通用的功能

Tag在Tags章节被涵盖,除了要讨论它和theme以及模板之间规定的关系,我们在这里不再更多地讨论它.作为替代,我们把注意力集中在几个重要的主题上:

- 模板装载 - WebWork如何装载模板
- 选择theme - 当编写你的结果页面时如何选择一个theme
- 扩展theme - 如何基于已有的theme创建自己的theme
- Themes - 详细概述WebWork包含的每个theme
    - simple theme - 一个最小的theme,包含了最少的辅助功能 ("bells and whistles")
    - xhtml theme - 使用了通用的HTML实践的缺省theme
    - css\ xhtml theme - 使用严格的CSS布局的对xhtml theme 重新实现
    - ajax theme - 一个基于xhtml theme 的theme,提供了高级AJAX特性

ajax theme

This page last changed on Apr 30, 2006 by scud.

ajax theme 扩展了 xhtml theme, 在它的父theme提供的每件事情之上提供了AJAX 特性.这个 theme 使用了两个流行的 AJAX/JavaScript 库: Dojo 和 DWR. 这些 AJAX 特性包括:

- AJAX客户端校验
- Remote form 提交支持 (最好和 submit 标签一起工作)
- 一个高级的 div 模板提供了动态装载部分HTML的功能
- 一个高级的 a 模板提供了加载并执行远端的JavaScript的能力
- 一个仅支持AJAX的 tabbedPanel 实现
- 一个"富"的 pub-sub 事件模型

Browser 兼容性

AJAX (当作一种技术) 使用不同浏览器(以及一些版本)之间变换的浏览器端脚本组件. 为了对开发者隐藏这些不同,我们利用了dojo 工具箱 (http://www.dojotoolkit.org). 下面的浏览器被dojo支持,并且任何AJAX theme创建的UI对于下面列出的那些浏览器都会行为一致:

- IE 5.5+
- FF 1.0+
- 最新的 Safari (最新的 OS 版本)
- 最新的 Opera
- 最新的 Konqueror

扩展 XHTML Theme

这个theme提供的包装行为几乎完全像xhtml theme提供的那样.唯一的不同就是 controlheader.ftl 模板有些不同:

```
<#include "/${parameters.templateDir}/xhtml/controlheader-core.ftl" />
<#if parameters.form?exists && parameters.form.validate?default(false) == true>
        <#-- can't mutate the data model in freemarker -->
    <#if parameters.onblur?exists>
        ${tag.addParameter('onblur', "validate(this);${parameters.onblur}")}
    <#else>
        ${tag.addParameter('onblur', "validate(this);")}
    </#if>
</#if>
    <td>
```

它通过检查 validate 属性是否设置为 true来提供 AJAX客户端校验 .如果是true, 每个HTML 标签的 onblur 事件都会设置一个校验请求.一些用户不喜欢这个 onblur行为,可能更喜欢一个更高级的 timer (例如 200ms) 在每次击键后被激活.如果你想要那么做你可以覆写这个模板来提供那类行为.

特别提示

因为这个theme里面的大多数的模板都能自解释,有一些模板可以提出来详细解释:

✓ 对于这个theme,强烈推荐你使用 head 标签. 浏览 ajax head template 了解更多信息. 没有它,大多数情况你就无法设置正确的AJAX支持.

- ajax head template
- ajax div template
- ajax a template

除了这些模板,让你自己熟悉WebWork和Dojo提供的ajax event system是很重要的.

ajax a template

This page last changed on May 01, 2006 by scud.

ajax a 模板,当用户点击一个超链接时,它被用来进行对服务器端的异步调用.当你需要从UI到应用程序进行信息通信时,这是有用的,它不需要整个页面被重新输出.例如从一个列表里移除一项.

preInvokeJS 属性用来确定是否指定的URL是否应该被调用,而且必须包含返回值为 _true_或 _false_的JavaScript.如果你想要调用一个JavaScript函数,使用 preInvokeJS='yourMethodName(data,type)' 的格式. 一个例子就是显示一个对话框来让用户仔细检查是否要从一个列表中移除一个用户.

记住: href 属性返回的内容必须是 JavaScript. 这些JavaScript 会被页面执行. 如果你仅仅是希望对于指定的topic发布一个事件,只需要简单地在你的action中不返回任何结果(或者什么也没有),并且使用 notifyTopics 属性来指定 topic 的名字.

关于在div标签和a标签之间使用 发布/订阅 交互的例子,请浏览 ajax div template 里面的例子.

ajax div template

This page last changed on May 01, 2006 by scud.

ajax div 模板提供了一个不同于其他theme方式的更有趣的div输出选择. 不仅仅是简单地输出一个 <div> 标签,这个模板依赖于Dojo Toolkit提供的高级AJAX特性. 当然div标签也能用于ajax theme之外,但它不是经常很有用. 浏览 div 标签来获取更多关于它提供了那些特性的信息.

特性

remote div有几个特性,其中一些可以和a标签以及ajax a template进行结合.这些用途是:

- 获取远端数据
- 初始化div的内容,在远端数据获取前
- 显示合适的错误和装载信息
- 定时周期刷新数据
- 监听事件并刷新数据
- JavaScript 控制支持

获取远端数据

最简单的方式使用div标签就是提供一个 href 属性.例如:

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"/>
```

这样做的结果是当HTML页面完全装载后,在浏览器里,指定的URL会以异步方式获取.那个URL返回的整个内容会被注入到div中.

初始化 Div

因为远端数据不是立刻装载进来的,有时候在远端数据获取之前需要一些占位符内容,这很有用.这些内容实际就是div元素的body.例如:

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239">
    Placeholder...
</ww:div>
```

如果你希望装载更多复杂的初始化数据,你可以使用action标签并设置 executeResult 属性:

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239">
    <ww:action id="weather" name="weatherBean" executeResult="true">
        <ww:param name="zip" value="97239"/>
    </ww:action>
</ww:div>
```

Loading 和 错误信息

如果你想在数据获取时或者数据不能被获取时显示一个指定的消息,你可以使用 errorText 和 loadingText 属性:

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"
        loadingText="Loading weather information..."
        errorText="Unable to contact weather server">
    Placeholder...
</ww:div>
```

刷新定时器

另外一个div 模板提供的特性就是基于定时器刷新数据的能力.使用 updateFreq 和 delay 属性,你可以指定多久定时器运作以及什么时候定时器开始运作(微秒).例如,下面的代码会在2秒延迟后每分钟更新一次:

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"
        loadingText="Loading weather information..."
        errorText="Unable to contact weather server">
        delay="2000"
        updateFreq="60000"
    Placeholder...
</ww:div>
```

监听事件

a标签 (指定为ajax a template) 时) 和div 标签支持一种 ajax event system, 提供了对topic进行广播事件的能力.你可以指定 *topic*来监听,只需要使用一个逗号分割的列表方式的 listenTopics 属性. 这意味着当一个通常是通过ajax a template 的topic发布时,href 属性指定的URL会被重新请求.

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"
        loadingText="Loading weather information..."
        errorText="Unable to contact weather server"
        listenTopics="weather_topic,some_topic">
    Placeholder...
</ww:div>
<ww:a id="link1"
      theme="ajax"
      href="refreshWeather.action"
      notifyTopics="weather_topic,other_topic"
      errorText="An Error ocurred">Refresh</ww:a>
```

JavaScript 支持

也有javascript函数来刷新内容或者停止/开始 刷新组件.对于id为"remotediv1"的remote div:

使用javascript开始刷新:

```
remotediv1.start();
```

使用javascript停止刷新:

```
remotediv1.stop();
```

使用javascript刷新内容:

```
remotediv1.bind();
```

JavaScript 例子:

为了解释这些概念,这里有一个例子. 假定你想要在运行时通过javascript改变一个div的url.. 这是你需要做的事情:
你需要做的就是添加一个JS函数,来监听从那个下拉框的id发布的JS事件.它会改变div的URL(添加这个id,这样正确的数据才能被获取),然后bind() AJAX div,因此它会刷新.

```
<ww:head theme="ajax" />

<script type="text/javascript">
    function updateReports(id) {
       var reportDiv= window['reportDivId'];
       reportDiv.href = '/../reportListRemote.action?selectedId='+id;
       reportDiv.bind();
    }
    dojo.event.topic.getTopic("updateReportsListTopic").subscribe(null, "updateReports");
</script>

<form ... >
<ww:select .... onchange="javascript: dojo.event.topic.publish("updateReportsListTopic",
this.value); " />

<ww:div id="reportDivId" theme="ajax" href="/.../reportListRemote.action" >
  Loading reports...
</ww:div>
</form>
```

ajax event system

This page last changed on May 01, 2006 by scud.

就像你可能已经在 ajax div template 和 ajax a template中看到的,WebWork和Dojo提供了一个优雅的方式来从浏览器上订阅和通知topic.使用Dojo作为很多的组件的基础的一个好处就是可以松散地连接成对的UI组件.有两个重要的属性: listenTopics 和 notifyTopics.

- 如果一个组件有一个 notifyTopics 属性, 那么在处理过程完成之后,一个信息将会被发布到这个属性提供的topic的所有名字上去(逗号分割).
- 如果一个组件有一个 listenTopics 属性, 那么当一个消息发布到一个topic名字上,如果这个topic名字是这个属性提供的名字的一部分(逗号分割的), 控件会进行自己标签特定的逻辑 (例如一个 DIV 标签会刷新自己的内容).

同时,你也可以使用JavaScript代码对一个topic名字来进行发布和订阅. 对于一个名字为 topic_name 的topic进行发布:

```
dojo.event.topic.publish("topic_name", "content");
```

topic_name 属性是必须的,content属性不是必须的,大多数元素会不使用这个属性进行触发.浏览 ajax div template 了解这种类型的交互的一个例子.

要订阅一个名字为 topic_name 的topic:

```
function doSomethingWithEvent(data) {
...
}
dojo.event.topic.getTopic("topic_name").subscribe(null, "doSomethingWithEvent");
```

subscribe 方法有2个参数,第一个是JavaScript对象变量(或者为null,如果函数不是从一个对象过来的),第二个是函数的名字,当topic收到一个事件时会调用的函数.

ajax head template

This page last changed on May 01, 2006 by scud.

ajax head 模板构建于 xhtml head template 之上,并提供了额外的 Dojo Toolkit JavaScript 包含,在 ajax a template, ajax div template 和 ajax tabbedPanel template 中被使用. 如果你希望使用 AJAX 特性,在你的 HTML `<head>` 块中,必须使用 `<ww:head theme="ajax"/>` 这个标签. head.ftl 的内容是:

```
<#include "/${parameters.templateDir}/xhtml/head.ftl" />
<script language="JavaScript" type="text/javascript">
    // Dojo configuration
    djConfig = {
        baseRelativePath: "<@ww.url includeParams='none' value='/webwork/dojo/'
encode='false'/>",
        isDebug: ${parameters.debug},
        bindEncoding: "${parameters.encoding}",
        debugAtAllCosts: true // not needed, but allows the Venkman debugger to work with the
includes
    };
</script>
<script language="JavaScript" type="text/javascript"
        src="<@ww.url includeParams='none' value='/webwork/dojo/dojo.js'
encode='false'/>"></script>
<script language="JavaScript" type="text/javascript"
        src="<@ww.url includeParams='none' value='/webwork/ajax/dojoRequire.js'
encode='false'/>"></script>
<script language="JavaScript" type="text/javascript"
        src="<@ww.url includeParams='none' value='/webwork/CommonFunctions.js'
encode='false'/>"></script>
```

如果你在让AJAX theme工作上遇到麻烦,你需要手动地包含上面的的JavaScript,改变 "isDebug: false" 到 "isDebug: true". 这会直接输出调试信息到屏幕上.

注意Dojo被配置使用在webwork.properties里指定的相同的字符集编码,通常是UTF-8.对于一个如何在AJAX theme里使用head标签的简单例子,只需要简单地在你的HTML里这样做:

```
<ww:head theme="ajax"/>
```

ajax submit template

This page last changed on May 06, 2006 by scud.

TODO: Describe the Ajax Submit template

ajax tabbedPanel template

This page last changed on May 06, 2006 by scud.

TODO: Describe the Ajax TabbedPanel template

css_xhtml theme

This page last changed on Apr 30, 2006 by scud.

css_xhtml theme 是WebWork缺省的theme.(译注:此句有误,xhtml才是WebWork默认的theme) 它除了提供所有simple theme 提供的所有基本元素外,还提供了这些额外的特性:

- 标准的两列 基于 CSS的布局,使用 `<div>` 方式来设置 HTML 标签 (form, textfield, select 等等)
- 为每个HTML 标签 设置label, 根据CSS的设置决定位置
- 校验 错误报告
- 纯JavaScript客户端校验 在浏览器上使用100%的JavaScript

包装 Simple Theme

css_xhtml theme 使用 **包装** 技术,在扩展Theme中提到过,xhtml theme也是这么做的. 同样的,对于理解HTML 标签如何使用一个标准的header和footer包装起来是很重要的.例如,查看一下textfield模板, text.ftl:
:

```
<#include "/${parameters.templateDir}/css_xhtml/controlheader.ftl" />
<#include "/${parameters.templateDir}/simple/text.ftl" />
<#include "/${parameters.templateDir}/css_xhtml/controlfooter.ftl" />
```

就像你能看到的那样, controlheader.ftl 和 controlfooter.ftl 模板包装在simple模板的外围.

CSS XHTML Theme Header

现在让我们看看 controlheader.ftl:

```
<#--
        Only show message if errors are available.
        This will be done if ActionSupport is used.
-->
<#assign hasFieldErrors = parameters.name?exists && fieldErrors?exists &&
fieldErrors[parameters.name]?exists/>
<div <#rt/><#if parameters.id?exists>id="wwgrp_${parameters.id}"<#rt/></#if> class="wwgrp">

<#if hasFieldErrors>
<div <#rt/><#if parameters.id?exists>id="wwerr_${parameters.id}"<#rt/></#if> class="wwerr">
<#list fieldErrors[parameters.name] as error>
    <div<#rt/>
    <#if parameters.id?exists>
     errorFor="${parameters.id}"<#rt/>
    </#if>
    class="errorMessage">
             ${error?html}
    </div><#t/>
</#list>
</div><#t/>
</#if>

<#if parameters.label?exists>
<#if parameters.labelposition?default("top") == 'top'>
<div <#rt/>
<#else>
<span <#rt/>
</#if>
<#if parameters.id?exists>id="wwlbl_${parameters.id}"<#rt/></#if> class="wwlbl">
    <label <#t/>
<#if parameters.id?exists>
```

```
            for="${parameters.id?html}" <#t/>
    </#if>
    <#if hasFieldErrors>
            class="errorLabel"<#t/>
    <#else>
            class="label"<#t/>
    </#if>
        ><#t/>
    <#if parameters.required?default(false)>
            <span class="required">*</span><#t/>
    </#if>
            ${parameters.label?html}:
    <#include "/${parameters.templateDir}/xhtml/tooltip.ftl" />
            </label><#t/>
    <#if parameters.labelposition?default("top") == 'top'>
    </div> <br/><#rt/>
    <#else>
    </span> <#rt/>
    </#if>
    </#if>

    <#if parameters.labelposition?default("top") == 'top'>
    <div <#rt/>
    <#else>
    <span <#rt/>
    </#if>
    <#if parameters.id?exists>id="wwctrl_${parameters.id}"<#rt/></#if> class="wwctrl">
```

css_xhtml theme里面HTML标签使用的header有点复杂. 不像 xhtml theme, 这个 theme 不支持 labelposition 属性. 作为替代,你的CSS规则可以定义如何设置布局.

也要注意 fieldErrors, 通常是因为 Validation 引发的, 在元素显示之前它限制在一个div块中.

CSS XHTML Theme Footer

下面是 controlfooter.ftl 文件的内容:

```
    ${parameters.after?if_exists}<#t/>
        <#lt/>
    <#if parameters.labelposition?default("top") == 'top'>
    </div> <#rt/>
    <#else>
    </span> <#rt/>
    </#if>
    </div>
```

特别提示

因为这个theme里面的大多数的模板都能自解释,有一些模板可以提出来详细解释:

- css_xhtml head template
- css_xhtml form template

css_xhtml form template

This page last changed on Apr 30, 2006 by scud.

css_xhtml form 模板几乎和 xhtml form template相同,包括对纯JavaScript客户端校验的支持.唯一的区别就是代替打印出开始和结束的 `<table>` 元素,没有输出任何元素.作为替代,对于独立的HTML标签的CSS规则假定来处理所有显示的逻辑.当然,就想前面标识的那样,客户端校验依然支持.

css_xhtml head template

This page last changed on Apr 30, 2006 by scud.

css_xhtml head模板非常类似 xhtml head template.唯一的区别是CSS包含进来,来指定css_xhtml theme的布局.
head.ftl 的内容是:

```
<link rel="stylesheet" href="<@ww.url value='/webwork/css_xhtml/styles.css' encode='false' />"
type="text/css"/>
<#include "/${parameters.templateDir}/simple/head.ftl" />
```

styles.css 的内容是:

```
.wwFormTable {}
.label {
    font-style:italic;
    float:left;
    width:30%
}
.errorLabel {font-style:italic; color:red; }
.errorMessage {font-weight:bold; text-align: center; color:red; }
.checkboxLabel {}
.checkboxErrorLabel {color:red; }
.required {color:red;}
```

Extending Themes

This page last changed on Apr 29, 2006 by scud.

有时你可能想要在一个已存在的theme里简单地覆盖一个模板(查看Template Loading)或者创建一个可选的模板.然而,其他时候你可能想要创建你自己的整个theme,特别是你计划为你的组织构建一个丰富的唯一的并且可复用的一套模板时.

有三种方法来创建一个新的theme:

- 白手起家,从头创建一个新的theme (**困难!**)
- 扩展一个已存在的theme
- 包装一个已存在的theme

我们永远不会推荐从头创建一个新的theme.更恰当地说,我们相信simple theme提供了为你扩展或者包装需要的足够的基本要素,从而来创建你自己唯一的theme.不管你选择了那个方法(经常情况下你最终两种方法都用了一些,因为它们不是互斥的),我们 **强烈** 建议你解压 WebWork的jar,并查看所有提供的theme的原有的模板.这会让你好好理解如何制造你自己的模板和theme.

包装一个存在的Theme

看一下xhtml theme,我们可以看到模板中广泛引用了 `wrapping` (包装)技术.例如,一个模板可能是这样的:

```
<#include "/${parameters.templateDir}/xhtml/controlheader.ftl" />
<#include "/${parameters.templateDir}/simple/xxx.ftl" />
<#include "/${parameters.templateDir}/xhtml/controlfooter.ftl" />
```

这个模板简单地使用一个header和一个footer包装了simple theme的已存在的模板.这是一种强大的方法,来在simple theme提供的基本的HTML元素外围增加额外的行为.

扩展一个存在的Theme

WebWork提供的theme基础设施也允许theme来 **扩展** 一个存在的theme. 这意味着一个theme可以包含一个 `theme.properties` ,其中有一个 `parent` 设置,包含了你想要扩展的theme的名字.例如,ajax theme就是用这种方式扩展xhtml theme的.

扩展一个theme,你不需要实现Tags里用到的每个模板.也就是说,你只需要实现你希望覆写的模板.其他的模板将从parent模板中装载.

Selecting Themes

This page last changed on Apr 29, 2006 by scud.

Theme可以使用不同的规则来选择,按照下面的顺序:

1. 特定标签上的 theme 属性
2. 一个标签外围的Form标签的 theme 属性
3. page 会话范围内的 以 theme 为名称的属性
4. request 会话范围内的命名为 theme 的属性
5. session 会话范围内的命名为 theme 的属性
6. application 会话范围内的命名为 theme 的属性
7. webwork.properties 内的 webwork.ui.theme 属性 (缺省是 xhtml)

对于这个顺序,有几个重要观念:

- 你可以简单地改变form的 theme 属性来覆盖整个表单的 theme设置. 这可以轻松地在几个选择的地方使用 ajax theme.
- 你可以基于用户的session来改变theme.如果用户能个性化他们的界面感观时,这可能很有用.
- 如果你想要改变整个应用的theme,调整 webwork.properties.

simple theme

This page last changed on Apr 30, 2006 by scud.

simple theme提供了"不加渲染"的HTML元素支持,被认为是最底层的结构,可以用来构建附加的功能或者行为(浏览扩展Theme了解更多信息). 例如,textfield标签输出HTML `<input />` 标签,不带额外的label,校验错误报告或者其他东西.如果你需要额外的行为,请浏览xhtml theme.

✓ xhtml theme 和 css\ xhtml theme 都是直接扩展了simple theme. 查看它们可以作为如何利用simple theme提供的基本要素的例子.

因为这个theme里面的大多数模板都是可以自解释的,有一些模板应该被提出来详细解释:

- simple head template

simple head template

This page last changed on Apr 30, 2006 by scud.

simple theme head 模板仅做了一件事情:就是打印一个 HTML <link/> 以便输出datepicker标签需要的CSS属性,保证它的正确显示.

simple head.ftl 模板的源码是:

```
<#if parameters.calendarcss?exists>
<link rel="stylesheet" href="<@ww.url
value='/webwork/jscalendar/${parameters.calendarcss?html}' />" type="text/css"/>
</#if>
```

Template Loading

This page last changed on Apr 29, 2006 by scud.

除非你有一些高级或者与众不同的要求,模板装载是非常容易理解的,简单的描述就是这样:

模板缺省情况下总是FreeMarker模板,装载方式和其他FreeMarker模板(例如你的结果页面)一样:首先搜索web应用程序然后搜索classpath.

模板(Template)和 Themes

模板是基于theme(查看选择theme)和模板目录来装载的.模板目录使用 `webwork.ui.templateDir` 属性,在webwork.properties里来定义(缺省是_template_).这意味着,缺省情况下,如果一个标签使用了 `ajax` theme,下面的两个位置将会被搜索(按照顺序):

1. 在web 应用中: /template/ajax/template.ftl
2. 在classpath里: /template/ajax/template.ftl

覆写模板

因为你需要的大多数的模板都包含在WebWork的jar(classpath)中,你可能会发现有些情况你需要覆写特定的模板来为你的应用提供独特的功能.例如,你可能希望改变select标签的输出,与其创建一个全新的模板并改变每个使用那个模板的标签,你可以覆写内置的 `select.ftl` 模板,只要从jar文件里面复制这个文件到一个新的 `/template/xhtml/select.ftl` 目录.

变更模板装载行为

有时不是从classpath和web 应用中,而是从别处装载模板可能是很重要的.例如,你可能希望从文件系统或者一个URL装载一个模板.这不仅对HTML Tags有用,可能对于任何你编写的result也有用.

你可以通过研究FreeMarker的文档来扩展FreeMarkerManager来达到这个目的.

可选的模板引擎

WebWork不仅仅提供了FreeMarker这个模板渲染引擎.我们几乎从来不推荐使用其他不是FreeMarker的模板,只是因为WebWork提供了已经使用FreeMarker编写的模板而不是使用JSP或者Velocity.然而,一些用户,特别是WebWork老版本(2.2之前)的高级用户可能会发现需要使用一个不同的模板引擎.

可选的模板引擎只为高级用户准备.请小心使用!

WebWork提供三种模板引擎,可以通过webwork.properties中的 `webwork.ui.templateSuffix` 属性控制:

- `ftl` (缺省) - 基于FreeMarker的模板引擎
- `vm` - 基于Velocity的模板引擎
- `jsp` - 基于JSP的模板引擎

如果你选择了使用 vm 或者 jsp ,你必须自己完全实现模板和theme,这是很大量的工作.

✓ 仅仅是因为你的视图不是使用FreeMarker编写的并不意味着你不能使用FreeMarker模板引擎.再次强调,我们 **强烈** 不推荐你从头编写自己的模板.更合适地说,我们推荐你学习一点FreeMarker,并且扩展已有的模板

xhtml theme

This page last changed on Apr 30, 2006 by scud.

xhtml theme是WebWork的缺省theme.它提供了所有simple theme提供的基本要素,加上一些附加的特性:

- 标准的两列表格布局,针对 HTML Tags (form, textfield, select等等)
- 每个HTML Tags的label,可以在左边或者上边,依赖 labelposition 属性的设置
- 校验 错误报告
- 纯JavaScript客户端校验 在浏览器上使用100% 的JavaScript

包装 Simple Theme

xhtml theme使用了 **包装** 技术,在 扩展Theme 里面提到过. 同样的,对于理解HTML 标签如何使用一个标准的header和footer包装起来是很重要的.例如,查看一下textfield模板, text.ftl:

```
<#include "/${parameters.templateDir}/${parameters.theme}/controlheader.ftl" />
<#include "/${parameters.templateDir}/simple/text.ftl" />
<#include "/${parameters.templateDir}/xhtml/controlfooter.ftl" />
```

就像你能看到的那样, controlheader.ftl 和 controlfooter.ftl 模板包装在simple模板的外围.也许你会感到惊奇,controlheader.ftl使用 ${parameters.theme} 的理由是帮助代码可以在ajax theme里面得到复用.现在,最好还是假设xhtml theme被使用.

XHTML Theme Header

现在让我们来看看 controlheader.ftl和controlheader-core.ftl (再次提醒,它们被分开是为了在ajax theme里面简单的进行复用)的内容:

```
<#include "/${parameters.templateDir}/xhtml/controlheader-core.ftl" />
    <td>
```

```
<#--
        Only show message if errors are available.
        This will be done if ActionSupport is used.
-->
<#assign hasFieldErrors = parameters.name?exists && fieldErrors?exists &&
fieldErrors[parameters.name]?exists/>
<#if hasFieldErrors>
<#list fieldErrors[parameters.name] as error>
<tr errorFor="${parameters.id}">
<#if parameters.labelposition?default("") == 'top'>
    <td align="left" valign="top" colspan="2"><#rt/>
<#else>
    <td align="center" valign="top" colspan="2"><#rt/>
</#if>
        <span class="errorMessage">${error?html}</span><#t/>
    </td><#lt/>
</tr>
</#list>
</#if>
<#--
        if the label position is top,
        then give the label it's own row in the table
-->
<tr>
```

```
<#if parameters.labelposition?default("") == 'top'>
    <td align="left" valign="top" colspan="2"><#rt/>
<#else>
    <td class="tdLabel"><#rt/>
</#if>
<#if parameters.label?exists>
    <label <#t/>
<#if parameters.id?exists>
        for="${parameters.id?html}" <#t/>
</#if>
<#if hasFieldErrors>
        class="errorLabel"<#t/>
<#else>
        class="label"<#t/>
</#if>
    ><#t/>
<#if parameters.required?default(false) && parameters.requiredposition?default("right") !=
'right'>
        <span class="required">*</span><#t/>
</#if>
${parameters.label?html}<#t/>
<#if parameters.required?default(false) && parameters.requiredposition?default("right") ==
'right'>
 <span class="required">*</span><#t/>
</#if>
:<#t/>
<#include "/${parameters.templateDir}/xhtml/tooltip.ftl" />
</label><#t/>
</#if>
    </td><#lt/>
<#-- add the extra row -->
<#if parameters.labelposition?default("") == 'top'>
</tr>
<tr>
</#if>
```

xhtml theme的header里面使用的HTML标签有点复杂.如果你仔细一点,你会看到逻辑产生了两种行为:一种是两列(two-column)格式,另外是两行(two-row)模式.通常两列方式是你需要的,因此这也是缺省的选项.当然,你通过修改 labelposition 参数为 "top"就可以使用两行(two-row)模式.

还要注意的就是 fieldErrors,通常是校验引发的,会在HTML表单元素的上面打印出来.一些用户可能希望在别处输出这些错误,例如在第三行.如果你希望把这些放在别处,覆写header是很简单的,允许你继续使用这个theme提供的其他特性.浏览模板装载来了解如何达到这个目的的更多信息.

XHTML Theme Footer

controlfooter.ftl 文件的内容:

```
${parameters.after?if_exists}<#t/>
    </td><#lt/>
</tr>
```

要注意的重要事情是表格的单元格和行被关闭了.在此之前,当然,你也要注意到一个特殊的 after 参数被检查了.注意这不是任何标签所支持的官方属性,如果你在任何模板引擎里使用FreeMarker Tags, Velocity Tags, 或者 param 标签,你可以添加一个 after 参数来存放任何你喜欢的内容,它会在simple theme模板输出后输出.这可以轻松的按照你的想法调整你的HTML表单.

特别提示

因为这个theme里面的大多数的模板都能自解释,有一些模板可以提出来详细解释:

- xhtml head template
- xhtml form template

xhtml form template

This page last changed on Apr 30, 2006 by scud.

xhtml表单模板设置所有其他xhtml theme表单元素的外围表格.出了创建外围的表格,开始和关闭模板之外,如果 validate参数设置为true,则开启 纯JavaScript客户端校验. 下面查看 *form.ftl*的内容:

```
<#include "/${parameters.templateDir}/xhtml/form-validate.ftl" />
<#include "/${parameters.templateDir}/simple/form.ftl" />
<table class="${parameters.cssClass?default('wwFormTable')?html}"<#rt/>
<#if parameters.cssStyle?exists> style="${parameters.cssStyle?html}"<#rt/>
</#if>
>
```

关闭的模板, form-close.ftl:

```
</table>
<#include "/${parameters.templateDir}/simple/form-close.ftl" />
<#include "/${parameters.templateDir}/xhtml/form-close-validate.ftl" />
```

xhtml head template

This page last changed on Apr 30, 2006 by scud.

xhtml head 模板扩展了 simple head template ,并提供了额外的CSS,以帮助渲染 xhtml theme 表单元素. *head.ftl*的内容是:

```
<link rel="stylesheet" href="<@ww.url value='/webwork/xhtml/styles.css' encode='false' />"
type="text/css"/>
<#include "/${parameters.templateDir}/simple/head.ftl" />
```

styles.css 的内容是:

```
.wwFormTable {}
.label {font-style:italic; }
.errorLabel {font-style:italic; color:red; }
.errorMessage {font-weight:bold; text-align: center; color:red; }
.checkboxLabel {}
.checkboxErrorLabel {color:red; }
.required {color:red;}
.tdLabel {text-align:right; vertical-align:top; }
```

Velocity Tags

This page last changed on Apr 19, 2006 by tin.

Velocity标签是WebWork提供的一般Tag的扩展. 只要简单的了解这些标签的访问方式: #wwxxx (...) ... #end (这里的xxx指WebWork支持的标签), 就可以马上开始使用了.

在WebWork 2.2中, 使用Velocity作为WebWork UI标签里的模版已经被移除了. 新的和遗留的2.2 UI标签已经使用FreeMarker来实现. 但是, 下面介绍的Velocity result type和Velocity directives将会被继续支持和开发下去. 默认的, WebWork的模版引擎将会使用下面介绍的Velocity语法调用FreeMarker UI标签(通过Velocity directives). 察看 模版载入 获取更多信息.

语法

例如, 在JSP中你可能这样创建一个form:

```
<ww:form action="updatePerson">
    <ww:textfield label="First name" name="firstName"/>
    <ww:submit value="Update"/>
</ww:form>
```

在Velocity中同样的form这样创建:

```
#wwform ("action=updatePerson")
    #wwtextfield ("label=First name" "name=firstName")
    #wwsubmit ("value=Update")
#end
```

块(Block)和内联(Inline)标签

你可能注意到了, 一些标签需要#end声明, 而其它的却不需要. 这是由于Velocity中的限制, 如果有 块 (block) 或者 内联 (inline) 标签在前面标签必须声明#end. 是这样的, 默认情况下所有的标签都是 内联 (inline) 的, 除了很少的几个, 例如 form 标签. 我们 **强烈** 建议你看看FreeMarker, 它在这部分和其它部分都给你提供了更多的自由.

Testimonials

This page last changed on Mar 30, 2006 by scud.

WebWork rocks! We use it for our Bug Tracking and for several of our clients. We have moved several sites from Struts to WebWork. I love it. Another site we work with for Survey Software is also moving off of Struts to WebWork. Everything is easier in WW, especially with the power of Interceptors!!!

Again... another site in 1 week with Site Mesh and WebWork. Its a blog community site. The one I manage... being a cyclist is the Cycling Community

Mike Porter, Architect, eSage Group
http://www.esagegroup.com

Two years ago we dediced to use WebWork instead of Struts because of it's technical superiority and it proved to be an excellent decision. WebWork is successfully used by productive customer applications running with WebLogic and Tomcat. A major project will be migrated to the newest XWork/WebWork versions in the next 6 months. Besides it's technical advantages, XWork/WebWork has a smart and extremely skilled developer team and a healthy community.

Lars Fischer, Project Manager, Compudata AG Switzerland
http://www.compudata.ch

WebWork is a very versatile web framework. After using solutions ranging from home-grown to Struts, WebWork is truly a breath of fresh air. XWork/WebWork not only used advanced techniques and technology, but brought concepts to the table that actually made development easier. These include built-in IOC, easy to use Spring integration, and null-property handling, and of course, type conversion.

I'm definitely looking forward to utilizing the newest features in my future projects. Keep up the good work!

Jay Bose, Sr. Engineer, Notiva Corporation
http://www.notiva.com

The Vault

This page last changed on Mar 30, 2006 by scud.

The Vault

What is the vault ?

The vault contains lots of unsorted problems, snippets, remarks and ideas, mostly found in posts in the WebWork Forum. You can think of this page as a place to store content before it is analyzed and put in the official docs, cookbook or FAQ. It's raw, unpolished, and can contain typos. If you can't find it in the docs, you can take a look in here (or use the search in the forum). It also gives an idea of what problems ww users encounter in their search for the holy grail.

Feel free to 'take' issues from this page and move them to an appropriate page.

Under construction
Tryout - if this is a bad idea, feel free to tell me so. It would be great if we could have some sort of notifier in the forum to inform us when an interesting post arrives.

Migration to 2.2

Note about upgrading your WW tags to 2.2

- http://forums.opensymphony.com/thread.jspa?threadID=13870&tstart=0

Just a quick fyi that caused issues for us... In 2.2 when extending UIBean evaluateExtraParams() no longer takes parameters. (Our parameters were not working until we realized that evaluateExtraParams was not being invoked due to the change.) In 2.1 evaluateExtraParams was passed the stack IIRC.

Not sure if this is documented anywhere but it caused us quite a headache when migrating from 2.1 to 2.2.

Where did the velocity templates go ?

- http://forums.opensymphony.com/thread.jspa?threadID=13108&tstart=0

checking cvs log, it seems that Pat has removed it with comment as follows:

removing velocity macro implementations - they are so old and out of date now it will cause more harm than good keeping them around

Best practices

Should I make an action or rather link to a .jsp page ?

- http://forums.opensymphony.com/thread.jspa?threadID=14424&tstart=0

If you put an action configuration in your xwork.xml but don't put an action
class attribute, it will default to using the ActionSupport class, which
just returns SUCCESS.

Type conversion

Type conversion for a static inner class ?

- http://forums.opensymphony.com/thread.jspa?threadID=15060&tstart=0

Collection_inner property name = your.package.ClassName$StaticInnerClassName (notice the dollar sign
between the fully qualified Class name and the static inner class)

Migration to Spring IoC

How can we migrate ServletRequestAware, ServletResponseAware and ValidationAware to Spring bean definitions?

- http://forums.opensymphony.com/thread.jspa?threadID=13551&tstart=0

I tink just the normal way would work. The action will now be created by spring, and if the
ServletConfigInterceptor is in the stack and the action implementation implement those interface, webwork
should set the appropriate methods.

Freemarker, Velocity, JSP

How to disable Velocity template caching ?

- http://forums.opensymphony.com/thread.jspa?threadID=15405&tstart=0

The "wwclass" was the resource loader I need to change.
Here's the lines from Velocity.props:

```
wwfile.resource.loader.cache=false
wwclass.resource.loader.cache=false
```

(http://jroller.com/page/gigix?entry=velocity_performance_issue_solved)

How to catch Freemarker exceptions (the ugly big yellow pages)

- http://forums.opensymphony.com/thread.jspa?threadID=13829&tstart=0

FreeMarker delegates exception handling to so-called TemplateExceptionHandlers. By default the
HTML_DEBUG_HANDLER is used (this one generates the yellow error page), but you can easily specify a
different exception handler by using the setTemplateExceptionHandler method of your configuration object.

FM provides some simple handlers you can use, in your case I recommend the RETHROW_HANDLER (http://freemarker.org/docs/api/freemarker/template/TemplateExceptionHandler.html).

My ww:include actions aren't processed

- http://forums.opensymphony.com/thread.jspa?threadID=13479&tstart=0

Did you declare the ww tag library in your includes ? If I'm not mistaken, jsp includes work different from ww includes (ww process the pages before including them).

I'm using ww:urlHelper in combination with the c:out tag, but it won't work !

- http://forums.opensymphony.com/thread.jspa?threadID=14312&tstart=0

if the url with &amp is used in the value attribute it will not work properly unless the escapeXml is set to false.

Validation

How to disable validation on the first page load (to show a in input form)?

- http://forums.opensymphony.com/thread.jspa?threadID=14634&tstart=0

If you define your own stack you can exclude methods from validation by adding the excludeMethods param to both Validation and Workflow. Assuming you are using the defaultStack the the methods input, back and cancel will be ignored. Change default to action!input and declare the input method to return INPUT.

When using clientside validation, DWR states it gets No Data From Server

- http://forums.opensymphony.com/thread.jspa?threadID=14135&tstart=0

When using firefox if a user has focus on an input control and then clicks the submit button i get a DWR error stating that there was no data from the server.
I searched around on this group and did not see a solution. Via the DWR mailing list I found a post fromi Joe Walker stating that a workaround was to register a custom DWR error handler ..
http://getahead.ltd.uk/dwr/browser/engine/errors

Why can't I use && (=and) in my validation rules ? || (=or) does work !

- http://forums.opensymphony.com/thread.jspa?threadID=15576&tstart=0

You can't use && directly since the validation files have to be wellformed xml. The correct syntax would be &amp ;&amp ; (without the spaces)

Ajax, Javascript

Datepicker i18n problems

- http://forums.opensymphony.com/thread.jspa?threadID=14132&tstart=0

DatePicker might have some problems in certain localised enviroments, because some of the language files included in DHTML / JavaScript Calendar are not valid (don't have some variables, for example WEEKEND definition).

Is simple solution to overide files like:
http://localhost:8080/showcase/webwork/jscalendar/lang/calendar-(your language).js

The only way to fix that, is correct calendar-(your language).js...

Already fixed for 2.2.1, available via CVS HEAD:

- sv
- pl
- cn

Application servers

WebWork hot-redeployment problem with Tomcat 5.5

- http://forums.opensymphony.com/thread.jspa?threadID=14553&tstart=0

Try to adjust your Tomcat context settings...

```
<Context docBase="${catalina.home}/webapps/YOURWEBAPP"
         antiResourceLocking="true" antiJARLocking="true">
...
</Context>
```

Deploying WebWork on Glassfish

- http://forums.opensymphony.com/thread.jspa?threadID=17532&messageID=34365#34365

One of my co-workers was deploying a forum application that uses WebWork code. He ran into some deployment and execution issues, related to the sercurity policy. If you want to deploy WebWork apps on GlassFish, you may want to read http://blogs.sun.com/roller/page/paulsen?entry=configuring_the_security_manager_in

Browser

Different version of a browser co-existing together

- http://blog.dojotoolkit.org/2005/12/01/running-multiple-versions-of-firefox-side-by-side
- http://labs.insert-title.com/labs/Multiple-IEs-in-Windows_article795.aspx
- http://www.skyzyx.com/archives/000094.php

Tutorial

This page last changed on May 22, 2006 by tin.

此教程还在进一步编写中
此教程还在编写完善中. 谢谢您耐心的等待.

1. 入门指南
2. 为你的web应用创建一个webapp项目结构
3. 第一课: 你的第一个Action
4. 第二课: 为你的Action提供一些数据
5. 理解results
6. 理解拦截器
7. 理解控制反转(IoC)
8. 理解校验
9. 理解标签库
10. Portlet教程

增补:

1. 在Eclipse中设置Tomcat

旧的教学指南 (它们虽然比较旧, 但是概念还是同样的):

1. 第一课: 设置Web应用
2. 第二课: An html form with no data
3. 第三课: An html form with data
4. 第四课: An html form with data, without getters or setters
5. 第五课: 视图 (JSP, Velocity, Freemarker)
6. 第六课: 拦截器

Basic configuration and your first action - Hello WebWorld

This page last changed on Apr 08, 2006 by dogg.

你的第一个action - Hello WebWorld

一个action是一个(particular)特殊的URL请求时执行的一段代码. 再actions执行后, a result visually displays the outcome of whatever code was executed in the action. A result is generally an HTML page, but it can also be a PDF file, an Excel spreadsheet, or even a Java applet window. In this book, we'll primarily focus on HTML results, because those are most specific to the Web.

When you submit an html form using WebWork, the form is sent to a Java class that you write yourself, not to a JSP page. These classes are called WebWork actions. The html form typically looks like: <form action="foo.action">.

在模型-视图-控制器方式中,WebWork的action是控制器的一部分, 留给JSP页面的是他们最擅长的工作: 创建视图. (如果你不知道什么是模型-视图-控制器, 不要担心.)

Suppose you want to create a simple "Hello, World" example in which a message is displayed whenever a user goes to a URL such as http://localhost/helloWorld.action. Because you've mapped WebWork's servlet to *.action, you need an action named helloWorld. To create the "Hello, World" example, you need to do three things:

1. 创建一个action类: HelloWorld.
2. 创建一个result: hello.jsp.
3. 配置action和result.

代码

```
package example.helloworld;
import com.opensymphony.xwork.Action;
import java.util.*;

public class HelloWorld implements Action {
private String message;

 public String execute() {
   message = "Hello, WebWorld!\n";
   message += "The time is:\n";
   message += DateFormat.getDateInstance().format(new Date());;

   return SUCCESS;
 }

 public String getMessage() {
   return message;
 }

}
```

粘贴此代码到文件webapp/hello.jsp中:

```
<%@ taglib prefix="ww" uri="webwork" %>
 <html>
   <head>
```

```
        <title>Hello Page</title>
      </head>
    <body>
        The message generated by my first action is:
        <ww:property value="message"/>
    </body>
  </html>
```

如下编辑xwork.xml文件, 添加helloWorld action和something called an interceptor to the default package. Read more: xwork.xml

```
    <!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
    "http://www.opensymphony.com/xwork/xwork-1.0.dtd">
      <xwork>
        <include file="webwork-default.xml"/>

        <package name="default" extends="webwork-default">
          <!-- Include webwork defaults (from WebWork JAR). -->
          <default-interceptor-ref name="completeStack"/>

           <action name="helloWorld"
              class="example.helloworld.HelloWorld">
              <result name="success">hello.jsp</result>
           </action>
        </package>
      </xwork>
```

不要忘记

编译你的action到 webapp/WEB-INF/classes下, 必须重新启动你的web应用.

现在,试着: 在你的浏览器地址栏中输入url http://localhost/helloWorld.action ,查看发生了什么. 你应该看到页面输出"Hello, WebWorld!".

代码如何工作

上述四个文件在一起是像这样工作的.

- 你的浏览器请求url http://localhost/helloWorld.action, 发送到你的web应用服务器.
- 应用服务器接受此请求helloWorld.action. 查看webapp/WEB-INF/web.xml中的配置, 它发现所有的 *.action (这是缺省配置) requests are to be handed off to com.opensymphony.webwork.dispatcher.FilterDispatcher. Essentially, the request is handed to WebWork now.
- WebWork 查看在 xwork.xml 中action名为 "helloWorld"的配置. There it finds that this corresponds to the class "HelloWorld," instantiates it, 调用它的方法execute().
- execute() 返回 SUCCESS, WebWork 又一次在xwork.xml 中查看,装载返回值是SUCCESS 时的页面. 它找到页面"hello.jsp".
- 页面hello.jsp 处理(此<ww:property value="message" />标签调用HelloWorld.java中的方法 getMessage()) 和发送到浏览器.

结束语,就WebWork, 所有的html表单元素发送到一个action. 此action返回一个常数result-name,比如:SUCCESS, ERROR或INPUT. Based on the xwork.xml, a given result-name may produce a page (as in this example), another action, or some other web resource (image, pdf). In this example, the form contained no data.

Create a webapp project structure for your web application

This page last changed on Apr 10, 2006 by dogg.

开始一个空白的web应用

你能使用 webapps/build.xml 文件创建一个项目结构帮助你快速的开始.
只要运行'ant new'和根据提示信息进行.

```
new:
     [echo]
     [echo]                  +=============================================================+
     [echo]                  |              -- Create a new web application  --            |
     [echo]                  +=============================================================+
     [echo]

    [input] Enter the name of your new application [myapp]? ww-sample
     [echo] Creating 'ww-sample' web application...
     [copy] Copying 7 files to /Users/rainerh/projects/webwork/webapps/ww-sample
     [copy] Copying 1 file to /Users/rainerh/projects/webwork/webapps/ww-sample
     [echo]
     [echo]                  +=============================================================+
     [echo]                  |    -- Your Web Application was created successfully! --     |
     [echo]                  |                                                             |
     [echo]                  | Now you should be able to cd to your application and run:   |
     [echo]                  | > ant build -Dwebapp=ww-sample                              |
     [echo]                  +=============================================================+

BUILD SUCCESSFUL
```

此任务是在webapps目录下生成一个新的目录.

例如, 下面是使用'ant new'创建的名为 'ww-sample'的一个新的webapp项目:

```
webapps/
  ww-sample/
    src/
      java/
         com/opensymphony/webwork/example/HomeAction.java -- A simple action example
implementation
      webapp/
        index.jsp -- redirects to home.action
        WEB-INF/
          classes/
            webwork.properties -- Simple properties to use Spring and run webwork in devMode
            xwork.xml -- Basic action mapping sample with 1 action mapping
          pages/
            home.jsp -- The home.jsp referenced via the HomeAction
          applicationContext.xml -- blank Spring definition file. Add your Spring beans here.
          web.xml -- basic web.xml for webwork
```

你现在能重新创建一个基于webwork运行的项目结构.

你可能想查看其他一些webapp项目例子,或得到一些更高级的使用范例.请查看showcase, starter, 或者shopping-cart应用.

Getting Started

This page last changed on May 22, 2006 by tin.

序言

WebWork 是一个流行的，容易使用的 MVC 框架. 如果需要更多的关于WebWork项目的信息，请访问 WebWork项目主页.
此文档将有助于你开始使用WebWork，帮助你运行例子和演示程序,即使你不是一个有经验的java web应用开发人员. 不过,
使用WebWork开发,你也需要懂得一些相关技术.在深入理解WebWork之前，推荐您重新审视下列概念:

- Java
- Servlets, JSP,Tag Libraries
- JavaBeans
- HTML and HTTP
- Web 容器(例如Tomcat)
- XML

下载发行包和相关资源

下载Webwork发行包

你能从这里下载WebWork的所有发行包. 它包含 webwork.jar文件和全部文档，源代码，所有 必须的和可选的依赖库文件,
示例. 如果需要从WebWork源代码或CVS捡出重新编译WebWork包，请查看构建WebWork.

联机信息和项目资源

在网络上有很多的关于Webwork项目的资源和信息,下列链接能帮助你找到这些:

- 下载Webwork - 下载WebWork发行包
- WebWork论坛 - 此论坛由积极的开发者，代码捐献者以及重要用户组成.这里也是得到问题解答的最好的，最快的地方.
- Webwork邮件列表或浏览邮件存档或者张贴问题. All posts from the forum are posted to the mailing lists, as well as your posts to the list will make it to the forum.
- CVS - 浏览CVS和java.net网站的代码
- Webwork Wiki - Powered by Confluence, 专业的 J2EE wiki
- Webwork的Bug和问题提交 - Powered by JIRA:Bug & Issue Traking System
- OpenSymphony主页

Distribution Quickstart

概述

发行包包含下列目录结构:

```
dist/
```

```
docs/
lib/
src/
src/java/template/
webapps/
README.txt
build.properties
build.xml
ivy.xml
osbuild.xml
pom.xml
webwork-(VERSION).jar
webwork-(VERSION).zip
webwork-(VERSION)-src.jar
```

docs目录包含当前版本的Javadoc, 您正在阅读的文档,标签库文档 以及Junit的构建报告.

在dist目录包含WebWork的不同jar封装文件:

- webwork-nostatic-<version>.jar: 仅仅包含WebWork的非静态内容.
- webwork-static-<version>.zip: 包含WebWork必须的静态内容.

lib目录包含WebWork必需的以及可选的依赖包, organized in subdirectories to represent different optional configurations:

```
lib/
      ajax
      bootstrap
      build
      cewolf
      default
      fileupload
      fileupload-cos
      fileupload-pell
      hibernate
      jasperreports
      jfree
      pico
      plexus
      portlet
      quickstart
      sitemesh
      source
      spring
      tiger
      tiles
      velocity
      xslt
```

注意:WebWork运行时可选包不是必须要的. 如果你希望使用某些特性如JasperReport或Java 5 (Tiger) 的注解支持,就必须包含所选特性的支持包.

Webwork也将全部的源文件和JSP标签模版打包在其中.

使用QuickStart运行示例

WebWork提供了一个快速启动应用示例的方法叫做QuickStart. QuickStart 从本质上讲是很少几种技术的组合, including the possibility to run web applications such as the provided examples out of the box with a stripped Jetty container.

With that, running the demos is as easy as can be. You just need to call:

```
java -jar webwork.jar quickstart:<application-name>
```

from the distribution's top directory, whereby <application-name> is to be replaced by one of the subdirectory names under webapps/, representing the various supplied demo applications. So if you want to start the shopping cart example, just type on your command line:

```
java -jar webwork.jar quickstart:shopping-cart
```

After starting an example in that way, you will simply need to point your browser to http://localhost:8080/shopping-cart to view the results.

The packaged examples are:

blank	Not really an example, but a blank web application template for creating your own application
portlet	Demonstration of WebWork porlet integration (to be deployed in a Portal Server) - see Portlet Tutorial to read more about that
shopping-cart	Nice example application demonstrating various aspects of WebWork
showcase	Demonstration of all tag and AJAX features, along with other examples and some best practices
starter	Basic web application, meant as an easy starting point for experimenting with WebWork's features - one could call it "playground"

Read Quickstart documentation to learn more about how it works and how you can utilize it for your own applications.

See below for a more detailed description of how the blank application helps you to easily create your own WebWork-based applications.

Running Demos with Your Favorite Webcontainer

To Quickstart or not to Quickstart?
The previous section described the Quickstart mechanism for trying the supplied example webapps, but Quickstart is also a handy tool for development of your own WebWork based applications. We recommend you using the said mechanism unless you are familiar with standalone servlet containers and have good reasons to switch, such as preparing a production system etc.

In order to deploy WebWork applications and demos to your favorite servlet container (also called web container) such as Apache Tomcat or Caucho Resin, you will need to build war files from within the webapps directory. You will find an ant build file there which will provide you with an easy way to accomplish that.

To build a webapp war archive, simply run:

```
ant build -Dwebapp=XXX
```

where XXX is the name of the webapp you want to build. After the build is finished, the fully-built war file can be found in the dist directory. You may deploy this file to any servlet container.

For example:
ant build -Dwebapp=showcase
To deploy the built showcase.war to Tomcat, just place it into <TOMCAT_HOME>/webapps/ or use the Tomcat Manager Application to upload and deploy the war. After the war is deployed (and your server is started), point your browser to
http://SERVER:PORT/APPLICATIONNAME, according to your setup and the application you deployed.

你的第一个WebWork应用

Using the Blank Template

As said before, the webapps directory contains a blank web application template. It is meant to be left untouched, and instead, you will want to make a copy of it as a starting point for your own new WebWork-based application. Although you can perform such a copy operation yourself, there is a much easier way around: You will find an build.xml in the webapps directory, which if called with ant new will provide you with a fresh empty webapp with a name you will be prompted for.

Setting up from Scratch

Structure of your Web Application

The following illustrates how your web application should be set up. Copy the webwork-(VERSION).jar, all the *.jar files in /lib/default and any necessary optional *.jar files in /lib/(optional configuration) to your webapp/lib directory. If you need to customize your own templates (how HTML is rendered from webwork UI tags), copy the /src/java/template directory into your webapp/ directory. Your webapp should look similar to this:

```
/mywebapp/
/mywebapp/template/
/mywebapp/META-INF/
/mywebapp/WEB-INF/
/mywebapp/WEB-INF/classes/
/mywebapp/WEB-INF/lib/
/mywebapp/WEB-INF/lib/CORE&OPTIONAL *.jar
/mywebapp/WEB-INF/web.xml
```

Minimum Set of Libraries and Config Files

The following files are a minium requirement for your application.

Filename	Description
webwork.jar	WebWork library itself, found in distribution root directory
xwork.jar	XWork library on which WebWork is built
oscore.jar	OSCore, a general-utility library from OpenSymphony

ognl.jar	Object Graph Navigation Language (OGNL), the expression language used throughout WebWork
commons-logging.jar	Commons logging, which WebWork uses to support transparently logging to either Log4J or JDK 1.4+
freemarker.jar	All UI tag templates are written in Freemarker, which is also a good option for your views
rife-continuations by rife	for the continuations feature
web.xml	J2EE web application configuration file that defines the servlets, JSP tag libraries, and so on for your web application
xwork.xml	WebWork configuration file that defines the actions, results, and interceptors for your application

The library files (*.jar) needs to be copied to your /mywebapp/WEB-INF/lib/ directory. If you need optional functionalities requiring dependencies on optional jars, they need to be copied to this directory, too.

Spring as default IoC Container
Before WebWork 2.2, the builtin Inversion of Control (IoC) container was default. Since Spring has been found to do this job better, the container integrated in WebWork / XWork was deprecated. If you would like to follow our recommendation to use Spring as the default IoC container, please make sure to include all jars found in lib/spring in the WEB-INF/lib directory of your application.

web.xml:

Create the following web.xml file in [webapp]/WEB-INF. If you already have a web.xml file, just add the content of the <web-app> tag below to your existing <web-app> tag.

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <display-name>My WebWork Application</display-name>
  <filter>
    <filter-name>webwork</filter-name>
    <filter-class>com.opensymphony.webwork.dispatcher.FilterDispatcher</filter-class>
  </filter>

  <filter-mapping>
    <filter-name>webwork</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
  <!-- As of 2.2, Spring is the preferred IoC container rather than XWork,
       so you'll have to include the spring jars if you want to use
       Spring's IoC capabilities in WebWork. (Thanks Hani for commenting)
       If you want to use deprecated integrated IoC container instead, you may
       want to omit the following listener configuration.
  -->
  <listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>

  <!-- The following taglib directive would be needed if your servlet container would comply
       to Servlet Spec <= 2.2
  <taglib>
    <taglib-uri>/webwork</taglib-uri>
    <taglib-location>/WEB-INF/lib/webwork.jar</taglib-location>
  </taglib>
  -->
</web-app>
```

This registers `FilterDispatcher` to enable webwork functionality for your requests. The ContextLoaderListener will take care of setting up Spring as your IoC container for WebWork. The taglib entry will help you use Tags in your JSP pages.

Read more: web.xml

xwork.xml:

Create the following file `xwork.xml` in `[webapp]/WEB-INF/classes/`.

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.1.1//EN"
"http://www.opensymphony.com/xwork/xwork-1.1.1.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
        </package>
</xwork>
```

For now, this xwork.xml does only two things:

- It informs WebWork that it should import the configuration information from `webwork-default.xml`. (This file is located at the root of the `webwork.jar`, so it is sure to be found.)

- It defines a default package (with the <package> section) where WebWork elements like actions, results and interceptors are registered.

Read more: xwork.xml

Onward to the First lesson or return to the Webwork Start page

Lesson 1 - Setting up webwork in a web application

This page last changed on Mar 30, 2006 by scud.

Lesson 1: Setting up webwork in a web application

For this lesson, you need to have a Servlet container set up and know how to create a web application. If you don't, we suggest you learn about Apache Tomcat, which is a free Servlet container from the Apache Jakarta Project, or Resin, from Caucho Technology, which is free for noncommercial use.

Notation

Throughout these lessons, we'll assume that your web application root is the directory [webapp], and that your Java source files are kept in [src].

To install WebWork in a web application:

1. Download WebWork. The current version can be found at WebWork's home page. This tutorial is based on version 2.1.7.
2. Set up an empty web application. For example, if you are using Tomcat, this will have something like the following directories (the directory "webwork-lessons" is referred to as [webapp] in these lessons):

```
[the tomcat root directory]
\|_webapps
  \|_webwork-lessons
    \|_WEB-INF
      \|_classes
      \|_lib
```

3. Copy the required WebWork libraries to your web application:

- copy `webwork-2.1.7.jar` to `[webapp]/WEB-INF/lib` ,
- copy `lib/core/*.jar` to `[webapp]/WEB-INF/lib` (not to `[webapp]/WEB-INF/lib/core`).

1. Configure `[webapp]/WEB-INF/web.xml`, and create `[webapp]/WEB-INF/classes/xwork.xml` and `[webapp]/WEB-INF/classes/validators.xml`, as described below.

WebWork jar name

If you have a later version of WebWork than 2.1.7, the WebWork jar will not be named webwork-2.1.7.jar. Be sure to replace all occurrences of this jar's name below with the name of the jar you are using.

web.xml:

Create the following `web.xml` file in `[webapp]/WEB-INF`. If you already have a `web.xml` file, just add the content of the <web-app> tag below to your existing <web-app> tag.

```
<?xml version="1.0"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
```

```
<web-app>
        <display-name>My WebWork Application</display-name>
        <servlet>
                <servlet-name>webwork</servlet-name>
                <servlet-class>com.opensymphony.webwork.dispatcher.ServletDispatcher</servlet-class>
                <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
                <servlet-name>webwork</servlet-name>
                <url-pattern>*.action</url-pattern>
        </servlet-mapping>
        <taglib>
                <taglib-uri>webwork</taglib-uri>
                <taglib-location>/WEB-INF/lib/webwork-2.1.7.jar</taglib-location>
        </taglib>
</web-app>
```

This registers `ServletDispatcher` as a servlet, and maps it to the suffix `*.action`. We will go into this more in the section on Actions in the next lesson. WebWork's taglib descriptor allows WebWork tags to be used (see lesson 4.1).

Read more: web.xml

xwork.xml:

Create the following file `xwork.xml` in `[webapp]/WEB-INF/classes/`.

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
        </package>
</xwork>
```

For now, this xwork.xml does only two things:

- It informs WebWork that it should import the configuration information from `webwork-default.xml`. (This file is located at the root of the `webwork-2.1.7.jar`, so it is sure to be found.)

- It defines a default package (with the <package> section) where WebWork elements like actions, results and interceptors are registered.

Read more: xwork.xml

validators.xml:

Create a file `validators.xml` in `[webapp]/WEB-INF/classes/` with the following content:

```
<validators>
        <validator name="required"
                class="com.opensymphony.xwork.validator.validators.RequiredFieldValidator"/>
        <validator name="requiredstring"
                class="com.opensymphony.xwork.validator.validators.RequiredStringValidator"/>
        <validator name="int"
                class="com.opensymphony.xwork.validator.validators.IntRangeFieldValidator"/>
        <validator name="date"
                class="com.opensymphony.xwork.validator.validators.DateRangeFieldValidator"/>
        <validator name="expression"
                class="com.opensymphony.xwork.validator.validators.ExpressionValidator"/>
        <validator name="fieldexpression"
                class="com.opensymphony.xwork.validator.validators.FieldExpressionValidator"/>
        <validator name="email"
                class="com.opensymphony.xwork.validator.validators.EmailValidator"/>
        <validator name="url"
                class="com.opensymphony.xwork.validator.validators.URLValidator"/>
        <validator name="visitor"
                class="com.opensymphony.xwork.validator.validators.VisitorFieldValidator"/>
        <validator name="conversion"
                class="com.opensymphony.xwork.validator.validators.ConversionErrorFieldValidator"/>
</validators>
```

This file defines the validators used, for example, for validating html form fields.

Read more: Validation

All Set Up!

Restart your servlet container (for example, restart Tomcat), and your webapp should be ready for use as a skeleton WebWork application.

To test whether everything is working, create `[webapp]/test.jsp`:

```
<html>
<body>
Hello html world
<hr/>
<%
  out.println("Hello jsp world.");
%>
</body>
</html>
```

If you can load this file in your browser and see the two Hello messages, your web application is working.

Next Lesson

This page last changed on Mar 30, 2006 by scud.

Lesson 2: An html form with no data

In this lesson, we are going to create a JSP with a form which, when submitted, loads a different JSP page saying "Hello, WebWorld!". To do that, we are going to write our first WebWork `action`.

Background: what are actions?

In JSP programming, submitting a form typically loads another JSP page where the form is processed using request.getProperty(). The form html looks like: `<form action="foo.jsp">`.

When you submit an html form using WebWork, the form is sent to a Java class that you write yourself, not to a JSP page. These classes are called WebWork `actions`. The form html typically looks like: `<form action="foo.action">`.

In a Model-View-Controller approach, the WebWork action is part of the Controller, leaving to JSP pages what they do best: the View. (If you don't know what a Model-View-Controller is, don't worry about this.)

The code

These are typical steps for creating a form and its action:

1. Create a JSP page with a form that calls the action.
2. Create the action class.
3. Register the action in `xwork.xml`.
4. Create a JSP page that will display the result.
5. Compile the action class. If necessary, restart your webapp.

Watch out for typos!

If something doesn't work properly, the first thing you'll want to do in this lesson is check all the files for typos, both in the files themselves as well as in the file names. This is a common source of errors.

1. Create a JSP page with a form that calls the action

Past this code into file called `page02.jsp`.

```
<html>
<head>
        <title>A simple form with no data</title>
</head>
<body>
        <p>Click the button below to activate Form02Action.</p>

        <form action="form02.action" method="post">
                <p><input type="submit" value="Click me." /></p>
        </form>

</body>
</html>
```

This is a form with no entry fields, just a submit button. Notice that the form's action attribute doesn't point to a jsp page, but to something strange called `form02.action`. We'll soon see why.

2. Create the action class

We are now going to create a Java class that will be part of the Java package "lessons". It doesn't matter where you keep this and other .java files; for example, they could be in these directories (if you are using Windows):

```
c:
\|_java
  \|_src
    \|_lessons
```

In these lessons, the above "src" directory is referred to as [src].

All our Java classes will be compiled to [webapp]/WEB-INF/classes. You'll have to include all the [webapp]/WEB-INF/lib/*.jar files in your CLASSPATH in order to compile these classes.

Paste this code into a file [src]/lessons/Form02Action.java:

```
package lessons;

import com.opensymphony.xwork.ActionSupport;

public class Form02Action extends ActionSupport {
        String hello;

        public String getHello() {
                return hello;
        }


        public String execute() throws Exception {
                hello = "Hello, WebWorld!";
                return SUCCESS;
        }

}
```

3. Register the action in `xwork.xml`

Edit the xwork.xml file as shown below, adding the form02 action and something called an interceptor to the default package.

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
                <!-- Default interceptor stack. -->
                <default-interceptor-ref name="defaultStack" />

                <!-- 02 -->
                <action name="form02" class="lessons.Form02Action">
                        <result name="success" type="dispatcher">page02-success.jsp</result>
                </action>
        </package>
</xwork>
```

Read more: xwork.xml

4. Create a JSP page that will display the result

Paste this code into a file [webapp]/page02-success.jsp:

```
<%@ taglib uri="webwork" prefix="ww" %>
<html>
<head>
        <title>Success page for form with no data</title>
</head>
<body>

<ww:property value="hello" />

</body>
</html>
```

Try it

Don't forget to compile your action to [webapp]/WEB-INF/classes, and to restart your web application if necessary.

Go ahead and try it now: click the form submit button on page02.jsp and see what happens. You should see a page that says "Hello, WebWorld!".

How the code works

The above four files work together like this.

- You click the form submit button on page02.jsp, sending it to your web application server.
- The server receives the request for `helloWebWebWorld.action`. Looking in [webapp]/WEB-INF/web.xml, it sees that all *.action requests are to be handed off to `com.opensymphony.webwork.dispatcher.ServletDispatcher`. Essentially, the request is handed to WebWork now.
- WebWork looks in xwork.xml for an action named "form02". There it finds that this corresponds to the class "lessons/Form02Action," instantiates it, and calls its excute() method.
- execute() returns SUCCESS, and WebWork looks again in xwork.xml to see what page to load if SUCCESS is returned. It finds the page "form02-success.jsp".
- The page page02.jsp is processed (the `<ww:property value="hello" />` tag calls the getter `getHello()` of Form02Action) and sent back to the browser.

To sum up: with WebWork, all html forms are sent to actions. The actions return constants like SUCCESS to specify (via xwork.xml) what page to return.

In this example, the form contained no data. In the next lesson, we'll see how to send form data to an action. Since page02-success.jsp called a getter of the action, you might guess that the form fields are going to call setters. You'd be right.

Previous Lesson | Next Lesson

This page last changed on Mar 30, 2006 by scud.

Lesson 3: An html form with data

In this lesson, we will create a form in which you can enter your name. For example, if you enter "Bob" and click the submit button,
you'll get a page saying "Hello, Bob!". If you don't enter a name, you'll get a screen saying: "Hmm, you don't seem to have entered a name. Go back and try again please."

As before, we set everything up in four steps: create the form, create the action, register the action, and create the landing page (or in this case, pages).

1. Create the form

Paste this html into [webapp]/page03.jsp:

```
<html>
<head>
        <title>A simple form with data</title>
</head>
<body>
        <p>What is your name?</p>

        <form action="form03.action" method="post">
                <p><input type="text" name="yourName"></p>
                <p><input type="submit" value="Submit your name." /></p>
        </form>

</body>
</html>
```

2. Create the form action

Paste this code into [src]/lessons/Form03Action.java:

```
package lessons;

import com.opensymphony.xwork.ActionSupport;

public class Form03Action extends ActionSupport {

  String yourName;

  public void setYourName(String p_yourName) {
    yourName = p_yourName;
  }

  public String getYourName() {
    return yourName;
  }


  public String execute() throws Exception {
    if (yourName == null || yourName.length() == 0)
      return ERROR;
    else
      return SUCCESS;
    }

  }
```

3. Register the action in xwork.xml:

Edit [webapp]/WEB-INF/classes/xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
  <!-- Include webwork defaults (from WebWork JAR). -->
  <include file="webwork-default.xml" />

  <!-- Configuration for the default package. -->
  <package name="default" extends="webwork-default">
    <!-- Default interceptor stack. -->
    <default-interceptor-ref name="defaultStack" />

    <!-- 02 -->
    <action name="form02" class="lessons.Form02Action">
      <result name="success" type="dispatcher">page02-success.jsp</result>
    </action>

    <!-- 03 -->
    <action name="form03" class="lessons.Form03Action">
      <result name="success" type="dispatcher">page03-success.jsp</result>
      <result name="error" type="dispatcher">page03-error.jsp</result>
    </action>

  </package>
</xwork>
```

4. Create the success and error pages

Create [webapp]/page03-success.jsp:

```
<%@ taglib uri="webwork" prefix="ww" %>
<html>
<head>
        <title>Success page for form with data</title>
</head>
<body>

Hello, <ww:property value="yourName" />!

</body>
</html>
```

Create [webapp]/page03-error.jsp:

```
<html>
<head>
        <title>Error page for form with data</title>
</head>
<body>

Hmm, you don't seem to have entered a name. Go back and try again please.

</body>
</html>
```

Try it

Don't forget to compile your action to [webapp]/WEB-INF/classes, and to restart your web application if necessary.

Go ahead and try it now: click the form submit button and see what happens. Try it with and without entering a name.

How the code works

There are only two differences between this example and the previous lesson.

- When the action is called, its `setYourName()` setter is called with the contents of the form field named `yourName`.
- After the action has been called (which is when its execute() method returns), WebWork has two options. If ERROR is returned, WebWork will return page03-error.jsp; if SUCCESS, page03-success.jsp. Just as in the last lesson, the <ww:property> tag calls the action's getter (in this case, `getYourName()`).

Lesson 2 \- An html form with no data | Lesson 4 \- An html form with data, without getters or setters

This page last changed on Mar 30, 2006 by scud.

Lesson 4: An html form with data, without getters or setters

For the form field named "yourName" in the previous lesson, we also had to create the getters and setters getYourName() and setYourName() in the action, as well as the private variable yourName. With dozens of forms and hundreds of form fields, you'll be typing thousands of getters and setters. That can get old fast. In this lesson, we'll repeat the last lesson, but without any of that extra typing.

1. Create the html form

Use the same JSP form from the previous lesson, but change the form action to `page04.action`:

```
<html>
<head>
        <title>A simple form with data</title>
</head>
<body>
        <p>What is your name?</p>

        <form action="form04.action" method="post">
                <p><input type="text" name="yourName"></p>
                <p><input type="submit" value="Submit your name." /></p>
        </form>

</body>
</html>
```

2. Create the form action

Paste this code into [src]/lessons/Form04Action.java:

```
package lessons;

import com.opensymphony.xwork.ActionSupport;
import com.opensymphony.webwork.interceptor.ParameterAware;

import java.util.Map;

public class Form04Action extends ActionSupport implements ParameterAware {

  Map parameters;

  public Map getParameters() {
    return parameters;
  }

  public void setParameters(Map parameters) {
    this.parameters = parameters;
  }

  public String execute() {
    String[] yourName = (String[]) parameters.get("yourName");
    if(yourName == null || yourName[0] == null || yourName[0].length() == 0)
      return ERROR;
    else
```

```
        return SUCCESS;
    }
  }
```

Register the action in xwork.xml:

Edit [webapp]/WEB-INF/classes/xwork.xml:

```
  <!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
  "http://www.opensymphony.com/xwork/xwork-1.0.dtd">

  <xwork>
    <!-- Include webwork defaults (from WebWork JAR). -->
    <include file="webwork-default.xml" />

    <!-- Configuration for the default package. -->
    <package name="default" extends="webwork-default">
      <!-- Default interceptor stack. -->
      <default-interceptor-ref name="defaultStack" />

      <!-- 02 -->
      <action name="form02" class="lessons.Form02Action">
        <result name="success" type="dispatcher">page02-success.jsp</result>
      </action>

      <!-- 03 -->
      <action name="form03" class="lessons.Form03Action">
        <result name="success" type="dispatcher">page03-success.jsp</result>
        <result name="error" type="dispatcher">page03-error.jsp</result>
      </action>

      <!-- 04 -->
      <action name="form04" class="lessons.Form04Action">
        <result name="success" type="dispatcher">page04-success.jsp</result>
        <result name="error" type="dispatcher">page03-error.jsp</result>
        <interceptor-ref name="servlet-config"/>
      </action>

    </package>
  </xwork>
```

Create the success and error pages

We'll use the same error page, but create a slightly different success page page04-success.jsp. The only difference is the <ww:property> tag.

```
  <%@ taglib uri="webwork" prefix="ww" %>
  <html>
  <head>
          <title>Success page for form with data</title>
  </head>
  <body>

  Hello, <ww:property value="parameters.yourName" />!

  </body>
  </html>
```

Try it

Don't forget to compile your action to [webapp]/WEB-INF/classes, and to restart your web application if necessary.

Go ahead and try it now. Load `page04.jsp`, enter "Bob" in the text field, and click the form submit button. You should see `page04-success.jsp` saying "Hello, Bob!"

How the code works

You've probably figured out what is going on just from looking at the code.

Instead of a setter `setYourName()` setting a private variable `yourName` in the action, `setParameters()` magically extracts everything from the JSP `request` object and puts into a private local Map `parameters`. Then `execute()`, instead of looking for a `yourName` variable, is able to get the value of the "yourName" field from `parameters`. So far so good .

Back on the `page04-success.jsp` page, `<ww:property value="yourName" />` isn't going to work any more, because there is no `getYourName()` getter in the action. Instead, `<ww:property value="parameters.yourName" />` calls the `getParameters()` getter, and is able to get the value of the "yourName" field. Pretty neat!

We haven't covered how to handle radio buttons, checkboxes, and other strange html form fields. That involves dealing with the fact that every entry in the `parameters` Map is a String[]. We'll cover this in a later lesson.

Previous Lesson | Next Lesson

Portlet Tutorial

This page last changed on May 22, 2006 by tin.

索引

Step-by-Step 教程

介绍

这篇教程带你经历建立简单portlet应用的流程，使用Eclipse，JBoss Portal 2.2 和 WebWork Portlet框架.

安装Eclipse

这篇教程中，我们将会使用 Eclipse 3.1.1，可以从此处下载 http://www.eclipse.org

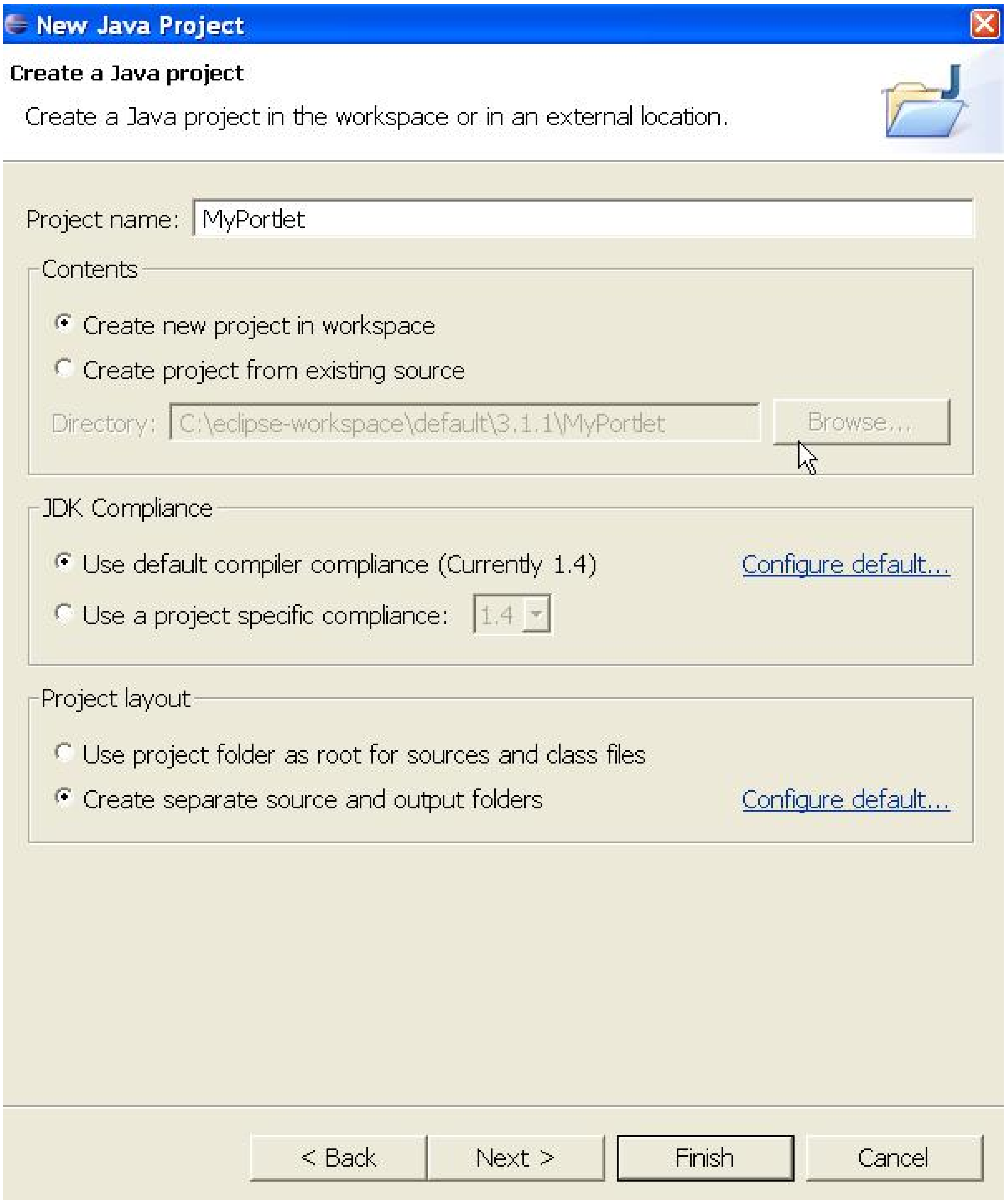
安装 JBoss Portal 2.2

JBoss Portal 2.2 可以在这里下载 http://www.jboss.com/products/jbossportal/downloads.

创建项目

一个Portlet应用与一般的web应用程序的打包方式基本相同，但是需要附加描述文件; portlet.xml. 本教程的第一步是在eclipse里面建立项目结构. 首先，我们创建这个 Java 项目本身，使用新建项目的向导. 我们命名项目为 'MyPortlet'. 确定选择了 "Create separate source and output folders (创建单独的源文件和输出目录)"选项，下一步，设置 'src' 源文件目录的输出目录(output folder)为'MyPortlet/webapp/WEB-INF/classes'. 讲讲方便我们在完成项目的时候导出本项目为 WAR 包.

New project wizard (新建项目向导)


New Java Project

Create a Java project

Create a Java project in the workspace or in an external location.

Project name: MyPortlet

Contents

- [●] Create new project in workspace
- [○] Create project from existing source

Directory: C:\eclipse-workspace\default\3.1.1\MyPortlet

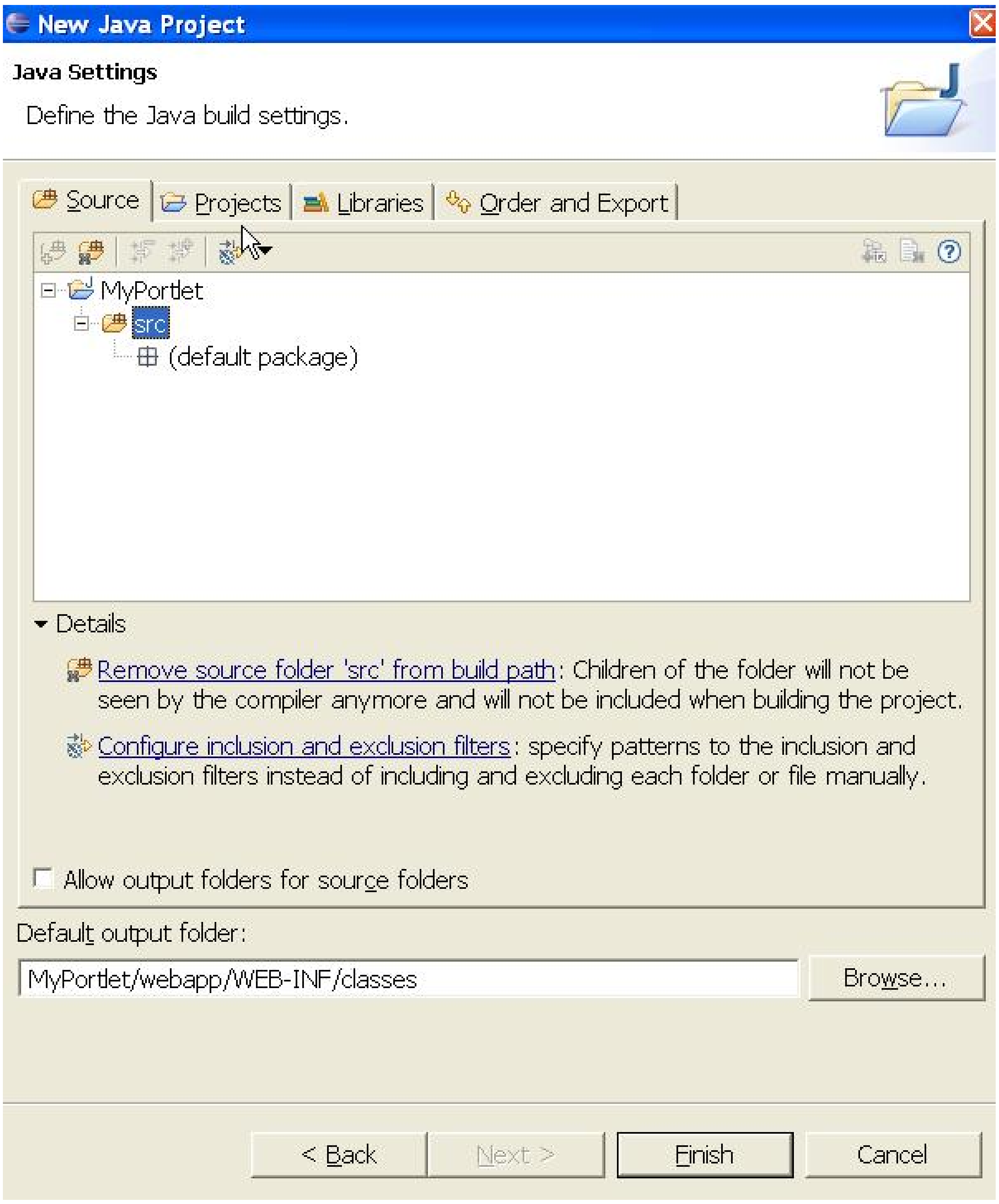
JDK Compliance

- [●] Use default compiler compliance (Currently 1.4)
- [○] Use a project specific compliance: 1.4

Project layout

- [○] Use project folder as root for sources and class files
- [●] Create separate source and output folders

New project wizard, cont (新建项目向导，续)


New Java Project

Java Settings

Define the Java build settings.

Remove source folder 'src' from build path: Children of the folder will not be seen by the compiler anymore and will not be included when building the project.

Configure inclusion and exclusion filters: specify patterns to the inclusion and exclusion filters instead of including and excluding each folder or file manually.

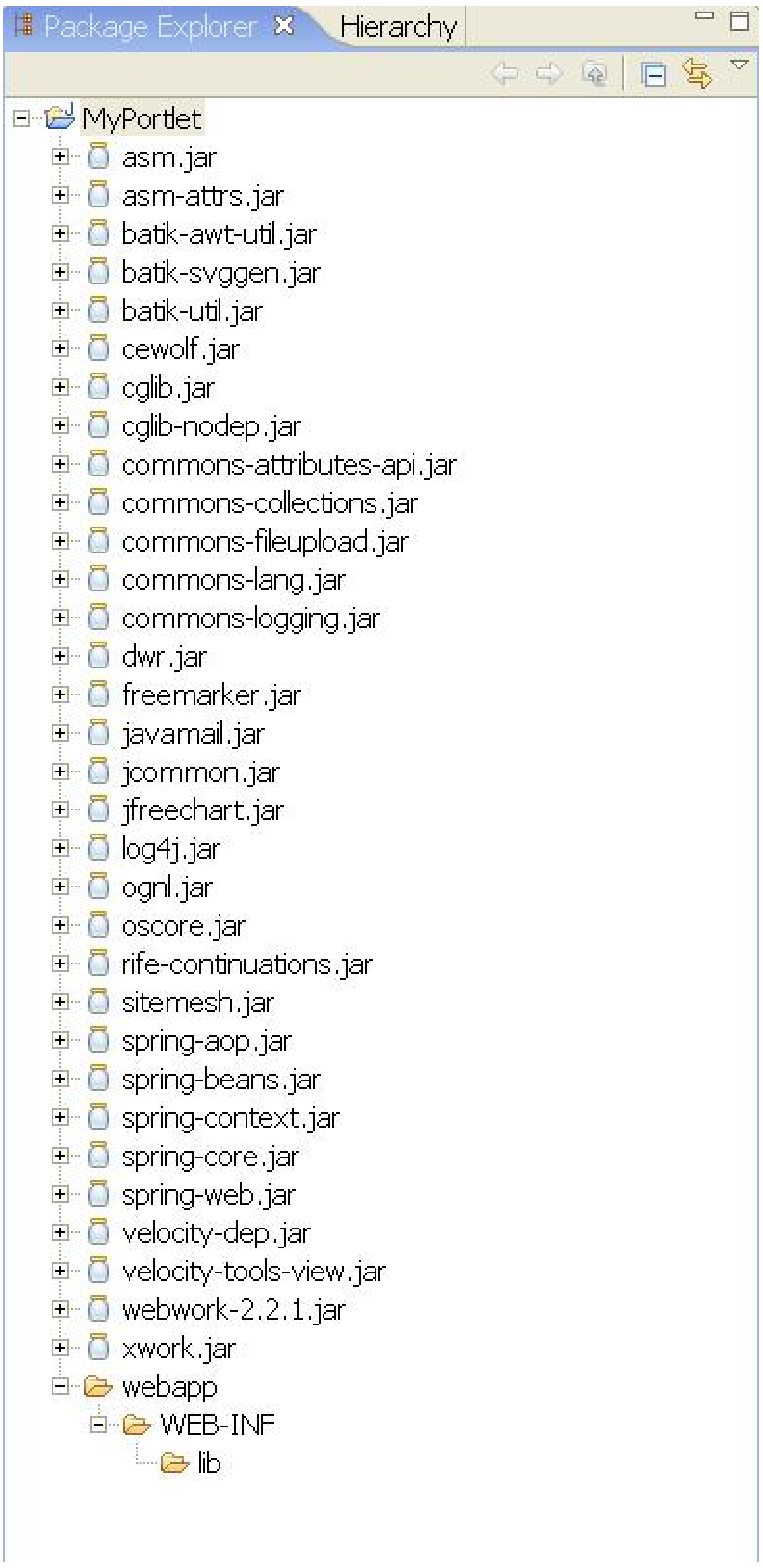
☐ Allow output folders for source folders

Default output folder:

MyPortlet/webapp/WEB-INF/classes

Classpath 设置

在 build 应用程序之前，我们需要添加一些需要的jar包到 build classpath 和 WEB-INF/lib 目录中. 首先，创建 WEB-INF/lib 目录，然后下载 WebWork 2.2.1 发布包并将起解压缩到本地硬盘. 找到下面屏幕截图中的jar包并将它们放到新创建的 WEB-INF/lib 目录中. 选择所有的jar包，点鼠标右键选择 "Build Path(Build路径) -> Add to Build Path(添加Build路径)". 现在你的本地项目应该看起来和屏幕截图中相同.



```
MyPortlet
  asm.jar
  asm-attrs.jar
  batik-awt-util.jar
  batik-svggen.jar
  batik-util.jar
  cewolf.jar
  cglib.jar
  cglib-nodep.jar
  commons-attributes-api.jar
  commons-collections.jar
  commons-fileupload.jar
  commons-lang.jar
  commons-logging.jar
  dwr.jar
  freemarker.jar
  javamail.jar
  jcommon.jar
  jfreechart.jar
  log4j.jar
  ognl.jar
  oscore.jar
  rife-continuations.jar
  sitemesh.jar
  spring-aop.jar
  spring-beans.jar
  spring-context.jar
  spring-core.jar
  spring-web.jar
  velocity-dep.jar
  velocity-tools-view.jar
  webwork-2.2.1.jar
  xwork.jar
  webapp
    WEB-INF
      lib
```

portlet.xml

Next thing we do is create a portlet.xml file in the WEB-INF folder. In this file, write the following:
我们下面要做的是在 WEB-INF 目录中创建一个 portlet.xml 文件. 在这个文件中, 写下如下内容:

```
<portlet-app version="1.0" xmlns="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd
http://java.sun.com/xml/ns/portlet/portlet-app_1_0.xsd">
  <portlet>
    <description xml:lang="EN">My very first WebWork Portlet</description>
    <portlet-name>MyPortlet</portlet-name>
    <display-name xml:lang="EN">My first WebWork Portlet</display-name>

    <portlet-class>com.opensymphony.webwork.portlet.dispatcher.Jsr168Dispatcher</portlet-class>

    <init-param>
      <!-- ####(view mode)#####(namespace). ###xwork######## -->
      <name>viewNamespace</name>
      <value>/view</value>
    </init-param>
    <init-param>
      <!-- ###########action## -->
      <name>defaultViewAction</name>
      <value>index</value>
    </init-param>

    <expiration-cache>0</expiration-cache>

    <supports>
      <mime-type>text/html</mime-type>
    </supports>

    <supported-locale>en</supported-locale>

    <portlet-info>
      <title>My very own WebWork Portlet</title>
      <short-title>WWPortlet</short-title>
      <keywords>webwork,portlet</keywords>
    </portlet-info>
  </portlet>
</portlet-app>
```

本 portlet.xml 文件设定这个 portlet 使用 com.opensymphony.webwork.portlet.dispatcher.Jsr168Dispatcher Portlet 实现. 它还告诉 Portlet 它会映射 视图 portlet 模式到 XWork 配置中的 /view 命名空间, 我们在建立(building)我们的 XWork action 的时候必须牢记它. 还有, 它告诉 portlet, 如果在 portlet 请求中没有找到一个 action 参数, 被调用的默认 action 是 "index" action, 它应该被放在 xwork 配置的 /view 命名空间里.

web.xml

WebWork Portlet 还需要你在 web.xml 描述中建立一些特殊的 servlet 和 filter 来打开对 WebWork 标签库和模版语言的支持, 因为它依赖于一些 Servlet API 的接口和类. 所以在WEB-INF 目录中创建 web.xml 文件, 并添加如下内容:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">
<web-app>
        <filter>
                <filter-name>webwork</filter-name>
                <filter-class>
                        com.opensymphony.webwork.dispatcher.FilterDispatcher
                </filter-class>
        </filter>

        <filter-mapping>
                <filter-name>webwork</filter-name>
                <url-pattern>/*</url-pattern>
        </filter-mapping>

        <listener>
```

```
                <listener-class>
                        com.opensymphony.webwork.portlet.context.ServletContextHolderListener
                </listener-class>
        </listener>


        <servlet>
                <servlet-name>preparator</servlet-name>
                <servlet-class>
                        com.opensymphony.webwork.portlet.context.PreparatorServlet
                </servlet-class>
        </servlet>

        <taglib>
                <taglib-uri>/webwork</taglib-uri>
                <taglib-location>/WEB-INF/lib/webwork-2.2.1.jar</taglib-location>
        </taglib>

</web-app>
```

HttpServletRequest/Response, and other Servlet API classes in the ServletActionContext that is used in many of the JSPs and templates.
FilterDispacher 保证 webwork jar 包中的风格样式表和 js 文件的URL可以被正确解析.
ServletContextHolderListener 是一个Servlet context listener(上下文监听器), 它存放了 servlet context (上下文) 的一个引用, 在派发(dispatch)到视图(例如JSP/ftl或velocity)之前初始化包括 HttpServletRequest/Response 和其它很多在JSP和模版引擎中会用到的ServletActionContext中的 Servlet API 类.

Hello World!

With these basic project structure, portlet.xml and web.xml in place, it's time to do the mandatory "Hello World" example, so let's create a place to store our JSP files. Create a WEB-INF/pages/view folder, and within this folder, create the file "helloWorld.jsp". In this file, we simply put:
拥有了基本的项目结构, portlet.xml 和 web.xml 都有了, 现在可以开始实现 "Hello World" 这个例子了, 我们先建立一个存放 JSP 文件的目录. 建立一个 WEB-INF/pages/view 目录, 然后在这个目录里, 建立一个 "helloWorld.jsp" 文件. 在这个文件中, 我们简单写下:

```
<H2>Hello world!</H2>
```

xwork.xml

At this point, it's time to prepare the xwork configuration file, xwork.xml. Create an empty file named xwork.xml in the root of the 'src' folder. In this file we put:
到了这里, 是准备 xwork 配置文件的时候了. 创建一个叫做 xwork.xml 的空文件, 放在 'src' 的根目录下.
在这个文件中, 我们放如下内容:

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<!DOCTYPE xwork PUBLIC
    "-//OpenSymphony Group//XWork 1.0//EN"
    "http://www.opensymphony.com/xwork/xwork-1.0.dtd">
<xwork>
        <include file="webwork-default.xml" />

        <package name="view" extends="webwork-portlet-default"
                namespace="/view">
```

```
                    <action name="index"
                            class="com.opensymphony.xwork.ActionSupport">
                            <result name="success">/WEB-INF/pages/view/helloWorld.jsp</result>
                    </action>
            </package>
    </xwork>
```

这里要明确一下，我们创建了一个命名空间为 view 的包(package)，我们的包继承自 webwork-portlet-default 包. 这个 webwork-portlet-default 包中包含了一些可以允许在 portlet 容器中运行 WebWork/XWork 的特殊结果类型(result type).

JBoss Portal 描述文件

除了常规的 portlet.xml 和 web.xml 表述文件, JBoss Portal 2.2 需要我们添加几个 JBoss 的特定描述文件. 这些描述文件的名称将根据我们的应用程序的上下文的根(context root) 的名字来命名, 它一般也是我们导出的 war 包的名字. 我们稍候将会创建一个叫做 MyPortlet.war 的 war 包, 所以 JBoss 描述文件被命名为 'MyPortlet-object.xml'. 我们在 WEB-INF目录中创建这个文件, 然后添加如下内容:

```
    <?xml version="1.0" encoding="UTF-8"?>
    <deployments>
            <deployment>
                    <if-exists>overwrite</if-exists>
                    <parent-ref>default</parent-ref>
                    <properties />
                    <page>
                            <page-name>MyPortlet Tutorial</page-name>
                            <properties />
                            <window>
                                    <window-name>MyPortletWindow</window-name>
                                    <instance-ref>MyPortletInstance</instance-ref>
                                    <region>center</region>
                                    <height>0</height>
                            </window>
                    </page>
            </deployment>
            <deployment>
                    <if-exists>overwrite</if-exists>
                    <instance>
                            <instance-name>MyPortletInstance</instance-name>
                            <component-ref>MyPortlet.MyPortlet</component-ref>
                    </instance>
            </deployment>
    </deployments>
```

然后, 我们需要另外两个文件, jboss-app.xml 和 jboss-portlet.xml 它们内容如下:

```
    <jboss-app>
       <app-name>MyPortlet</app-name>
    </jboss-app>
```

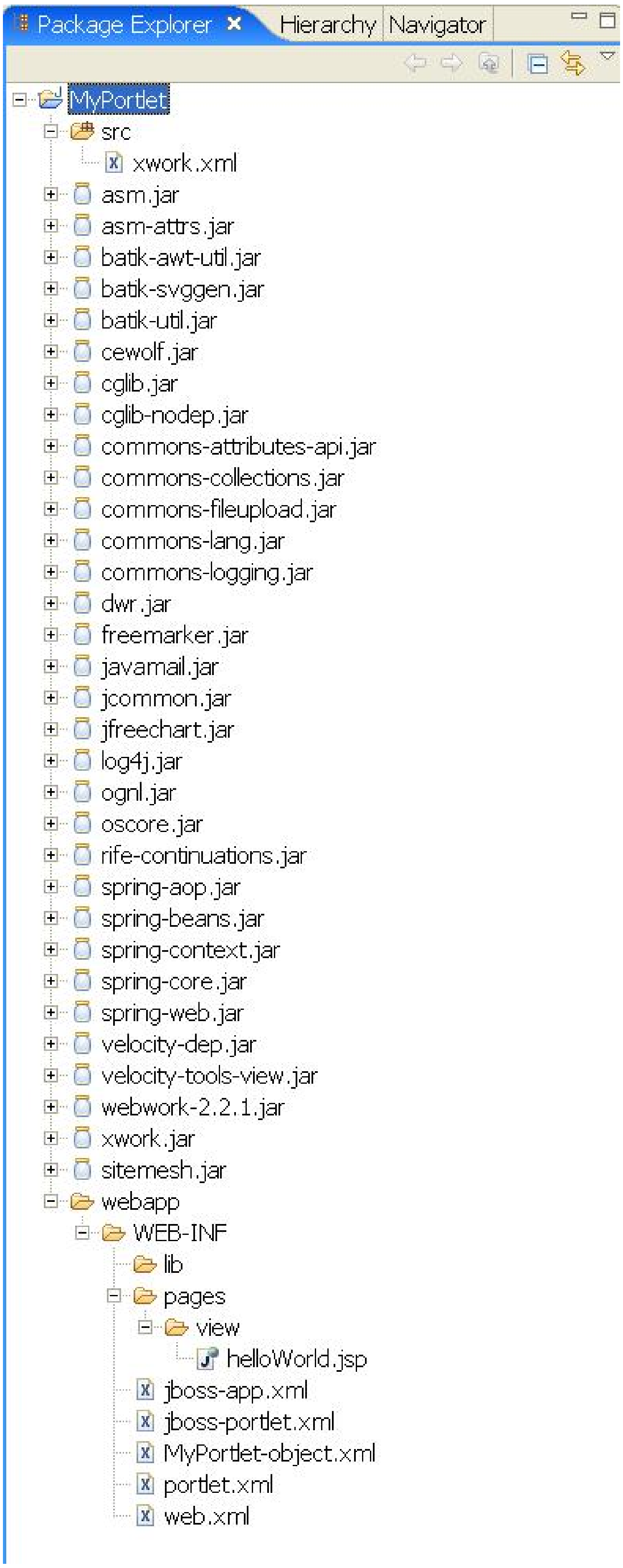
```
    <portlet-app>
       <portlet>
          <portlet-name>MyPortlet</portlet-name>
```

```
        <security>
        </security>
     </portlet>
  </portlet-app>
```

部署

现在我们的项目结构看起来是这样的:

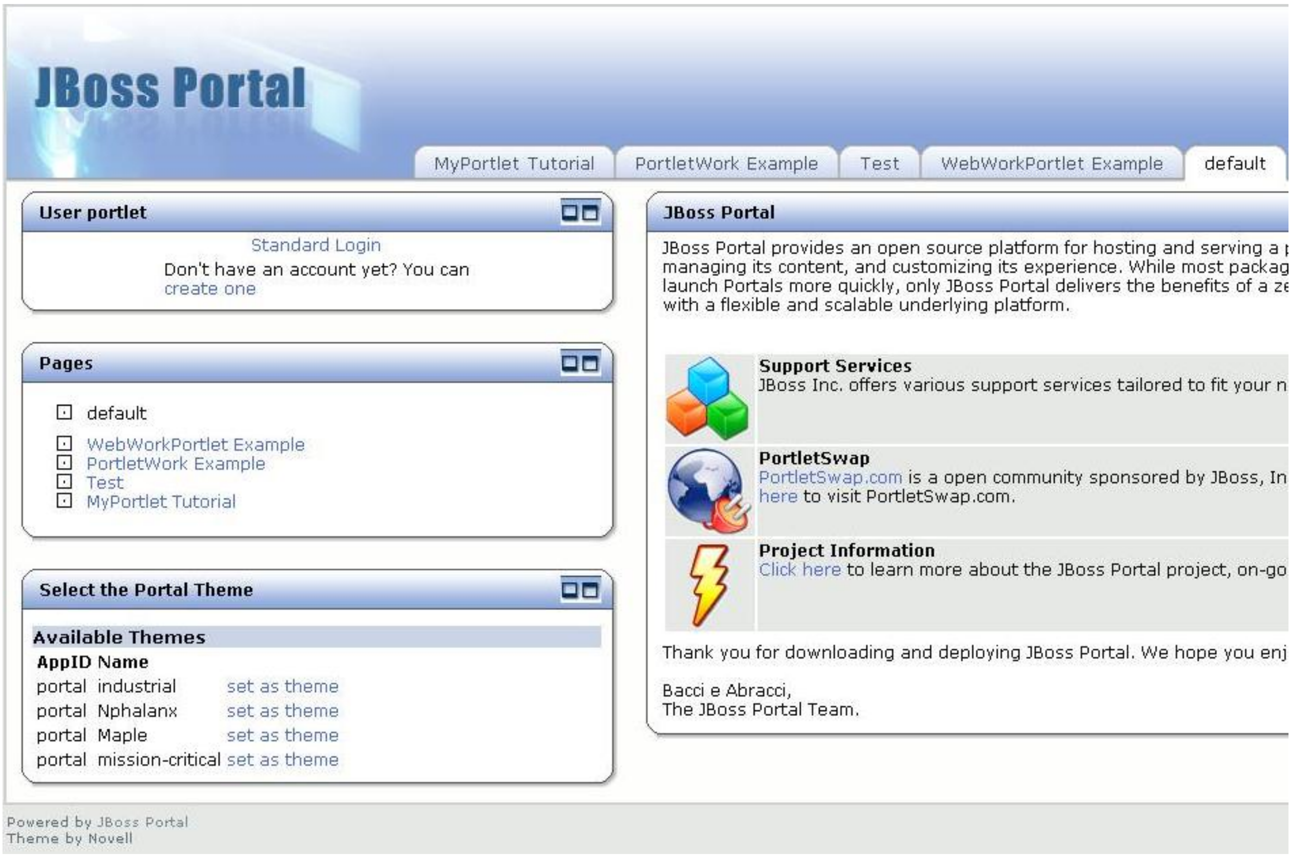
项目结构



```
MyPortlet
    src
        xwork.xml
    asm.jar
    asm-attrs.jar
    batik-awt-util.jar
    batik-svggen.jar
    batik-util.jar
    cewolf.jar
    cglib.jar
    cglib-nodep.jar
    commons-attributes-api.jar
    commons-collections.jar
    commons-fileupload.jar
    commons-lang.jar
    commons-logging.jar
    dwr.jar
    freemarker.jar
    javamail.jar
    jcommon.jar
    jfreechart.jar
    log4j.jar
    ognl.jar
    oscore.jar
    rife-continuations.jar
    spring-aop.jar
    spring-beans.jar
    spring-context.jar
    spring-core.jar
    spring-web.jar
    velocity-dep.jar
    velocity-tools-view.jar
    webwork-2.2.1.jar
    xwork.jar
    sitemesh.jar
    webapp
        WEB-INF
            lib
            pages
                view
                    helloWorld.jsp
            jboss-app.xml
            jboss-portlet.xml
            MyPortlet-object.xml
            portlet.xml
            web.xml
```

现在可以试验一下我们的不可思议的 HelloWorld portlet 了. 在Windows的一个浏览器窗口中, 我们选择 WEB-INF 目录然后将它以 zip 格式压缩为叫做 'MyPortlet.war' 的文件. 将这个 war 文件放到JBoss Portal的 server/default/deploy 目录中, 然后启动服务器. 默认情况下, JBoss Portal 的 URL 是 http://localhost:8080/portal, 所以我们用浏览器访问这个地址, 我们将会看到这个 portal 的首页, 这里你应该会看到一个 "MyPortlet Tutorial" 菜单项, 与下面的截图中的一相同. 当按下菜单链接, 你会得到一个奇妙的 "Hello World" 页面!

JBoss Portal 首页


JBoss Portal

MyPortlet Tutorial | PortletWork Example | Test | WebWorkPortlet Example | default

User portlet

Standard Login
Don't have an account yet? You can create one

Pages

- default
- WebWorkPortlet Example
- PortletWork Example
- Test
- MyPortlet Tutorial

Select the Portal Theme

Available Themes

AppID	Name	
portal	industrial	set as theme
portal	Nphalanx	set as theme
portal	Maple	set as theme
portal	mission-critical	set as theme

JBoss Portal

JBoss Portal provides an open source platform for hosting and serving a p
managing its content, and customizing its experience. While most packag
launch Portals more quickly, only JBoss Portal delivers the benefits of a ze
with a flexible and scalable underlying platform.

Support Services
JBoss Inc. offers various support services tailored to fit your n

PortletSwap
PortletSwap.com is a open community sponsored by JBoss, In
here to visit PortletSwap.com.

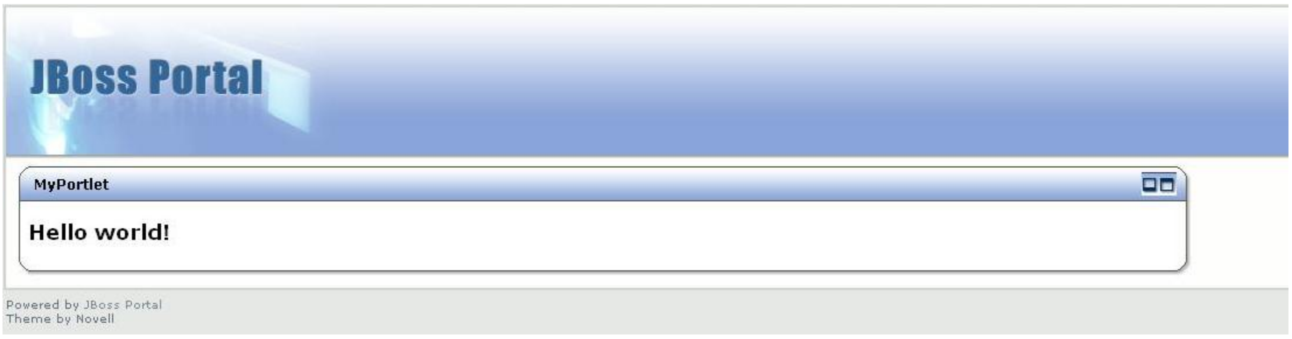
Project Information
Click here to learn more about the JBoss Portal project, on-go

Thank you for downloading and deploying JBoss Portal. We hope you enj

Bacci e Abracci,
The JBoss Portal Team.

Powered by JBoss Portal
Theme by Novell

MyPortlet portlet 页面


JBoss Portal

MyPortlet

Hello world!

Powered by JBoss Portal
Theme by Novell

下一步

Next, let's do something a bit more interesting, namely create a simple form and display a result page. Let's start by creating our JSP that displays our form. Create a new file, 'helloForm.jsp' in the WEB-INF/pages/view/ folder. We will use the WebWork tag library to build the form on our page. The form itself will ask the user for a first name and last name, something like this:
接着, 我们来做一些有趣的事, 也就是创建一个简单的 form 并显示一个结果页面. 我们先来创建一个 JSP 页面来显示这

个 form. 创建一个新的文件 'helloForm.jsp', 放在 WEB-INF/pages/view/ 目录下. 我们将会使用 WebWork 标签库在我们的页面上创建这个 form. 这个 form 会询问用户的名字和姓, 就像这样:

```
<%@ taglib uri="/webwork" prefix="ww" %>

<H2>Hi there! Please enter your name</H2>
<ww:form action="helloWorld" method="POST">
        <ww:textfield label="First name" name="firstName" value="%{firstName}"/>
        <ww:textfield label="Last name" name="lastName" value="%{lastName}"/>
        <ww:submit value="Say hello!"/>
</ww:form>
```

现在我们准备写一些 Java 代码, 不多, 就一点点. 我们在我们的 src 目录创建一个新的包, 我们命名它为 com.opensymphony.webwork.portlet.tutorial. 在这个包中, 创建一个 HelloWorldAction 类. 遵循一般的 WebWork 习惯, 这个类扩展自XWork框架的 ActionSupport 类, 我们还会添加一些属性来映射我们刚才创建的 JSP 中的 form 的属性:

```
package com.opensymphony.webwork.portlet.tutorial;

import com.opensymphony.xwork.ActionSupport;

public class HelloWorldAction extends ActionSupport {
        private String firstName;
        private String lastName;
        public String getFirstName() {
                return firstName;
        }
        public void setFirstName(String firstName) {
                this.firstName = firstName;
        }
        public String getLastName() {
                return lastName;
        }
        public void setLastName(String lastName) {
                this.lastName = lastName;
        }
}
```

We also need a JSP to display the processed input. We'll just use the old helloWorld.jsp and modify it a bit. As with helloForm.jsp, we import the WebWork tag library, and we use the ww:property tags to display the input from the form:
我们还需要一个 JSP 来显示 input 的处理结果. 我们还使用 helloWorld.jsp 然后修改一下. 与 helloForm.jsp 一样, 我们引入 WebWork 标签库, 然后我们使用 ww:property 标签来显示 form 中输入的内容:

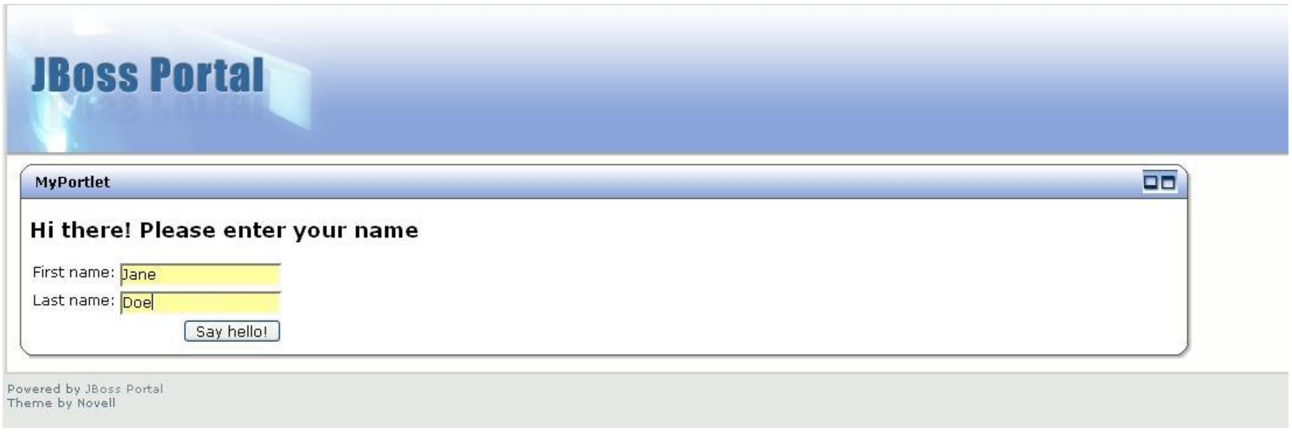
```
<%@ taglib prefix="ww" uri="/webwork" %>

<H2>Hello <ww:property value="firstName"/> <ww:property value="lastName"/></H2>
<p/>
<a xhref="<ww:url action="helloWorldInput"/>">Back to form</a>
```

重新部署

现在我们准备好重新部署我们的应用程序, 所以重新用 zip 格式压缩好一个 war 包并放到 server/default/deploy 目录. 现在 'MyPortlet Tutorial' 页面将会是这个样子:

Hello World form

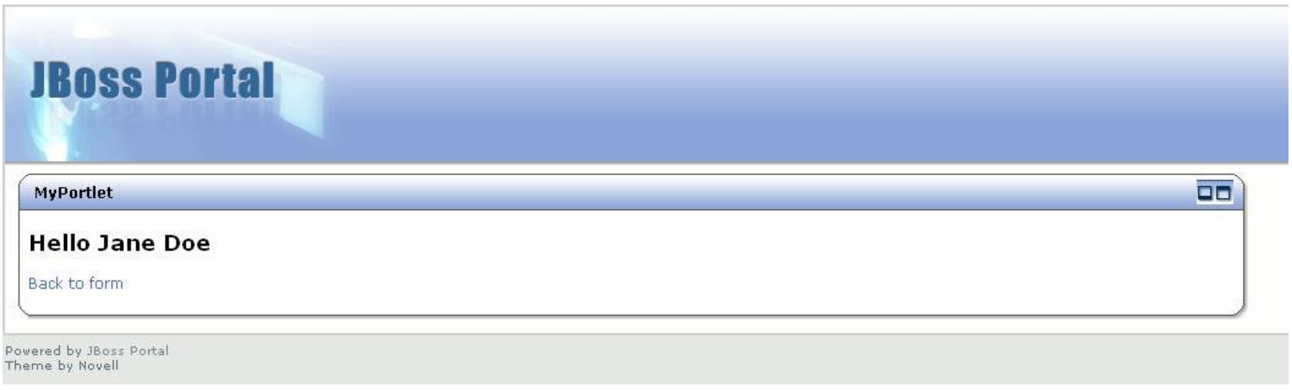


Hi there! Please enter your name

First name: Jane
Last name: Doe

输入一些信息，然后点击 "Say hello!" 按钮，然后你将会看到不错的个性化的 "hello" 信息：

个性化的 Hello World



Hello Jane Doe

Tiles Use

This page last changed on Mar 30, 2006 by scud.

Here is a simple example of tiles use with spring's TilesConfigurer. There is really no need for spring but tiles definitions must somehow be initialized. If you do not use Spring you could use the following context listener that is exatcly how Spring configures tiles definitions:

```
package com.opensymphony.webwork.views.tiles;

import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;

import org.apache.struts.tiles.DefinitionsFactoryConfig;
import org.apache.struts.tiles.DefinitionsFactoryException;
import org.apache.struts.tiles.TilesUtil;
import org.apache.struts.tiles.xmlDefinition.I18nFactorySet;

/*
* Modified from spring's source
*
* here's how a smaple web xml should look like:
* <web-app>
*   <context-param>
*     <param-name>tilesDefinitions</param-name>
*     <param-value>/WEB-INF/tiles.xml</param-value>
*   </context-param>
*
*   <listener>
*      <listener-class>com.opensymphony.webwork.views.tiles.TilesConfigurer</listener-class>
*   </listener>
* </web-app>
*
* To use the definitions specified you would use a dispatcher result (since
* tiles jsp is just another jsp) to render tiles view.
*/
public class TilesConfigurer implements ServletContextListener {

    private boolean initialized = false;

    public void contextInitialized (ServletContextEvent evt) {

        if (!initialized) {
            DefinitionsFactoryConfig factoryConfig = new DefinitionsFactoryConfig();
            factoryConfig.setFactoryClassname(I18nFactorySet.class.getName());
            factoryConfig.setParserValidate(true);
            factoryConfig.setDefinitionConfigFiles(evt.getServletContext().getInitParameter("tilesDefinitio
            try {
                TilesUtil.createDefinitionsFactory(evt.getServletContext(), factoryConfig);
            } catch (DefinitionsFactoryException e) {
                e.printStackTrace();
            }
            initialized = true;
        }

    }

    public void contextDestroyed (ServletContextEvent evt) {
    }

}
```

TutorialLesson05

This page last changed on Mar 30, 2006 by scud.

Lesson 5: Views

There are some different technologies that you could use as the view, i.e., to construct the user interface:

Lesson 5.1 - Java Server Pages

JSP is the common choice, because most Java web developers are already familiar with the technology. This lesson assumes you already have experience with Java Server Pages and demonstrates how you can use the WebWork features in JSP, mostly by using WebWork tags.

Go to lesson 4.1 (Currently named 4.x while documentation is being rewritten.)

Lesson 5.2 - Velocity

Velocity is a Java-based template engine that provides a simple, but powerful, template language that replaces JSP and allows for separation of concerns. This lesson assumes that you are already familiar with Velocity and teaches you how to use WebWork features from it.

Go to lesson 4.2 (Currently named 4.x while documentation is being rewritten.)

Lesson 5.3 - Freemarker

Designed for MVC pattern, Freemarker is another Java-based template engine that provides a powerful template language that replaces JSP, but can remain JSP-compatible with a JSP taglib support. This lesson teaches you how to use WebWork and Freemarker together.

Go to lesson 5.3 (Currently named 4.x while documentation is being rewritten.)

Previous Lesson | Next Lesson

TutorialLesson04-01

This page last changed on Mar 30, 2006 by scud.

Lesson 4.1: Using JSP as the View

When using JSP to render the views, you can choose to access the action's data using scriptlets or tags. Tags are the recommended approach.

Accessing Action Data through Scriplets:

Action data can be accessed through an object called Value Stack. The example below does the same thing as the result page of lesson 3's second example (Supplying Data to the Action), but using scriptlets:

```
<%@ page import="com.opensymphony.xwork.util.OgnlValueStack" %>
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1 - Lesson 3's example modified</title>
</head>
<body>

<%
OgnlValueStack stack = (OgnlValueStack)request.getAttribute("webwork.valueStack");
out.write("Hello, " + stack.findValue("person"));
%>

</body>
</html>
```

WebWork tags, however, are recommended over scriptlets. For instance, `<ww:property />` tags do exactly what the scriptlet above does, with a cleaner syntax and also handles the case where the Value Stack doesn't exist.

WebWork Tag Library:

We've already showed in lesson 3's example how to access an action's property using tags. This section describes and exemplifies the use of the WebWork Tag Library, which can be divided in seven categories:

- **Common tags**: the most frequently used, basic tags;
- **Componentisation tags**: foster componentisation within your views;
- **Flow control tags**: govern the flow of control within the JSP page;
- **Iteration tags**: iterate over elements and manipulate iterable objects;
- **UI tags**: generate HTML form fields and controls;
- **VUI tags**: volunteers needed to write this part;
- **Internationalisation tags**: internationalise your views.

Common tags

<ww:property />	Gets the value of a result attribute. If the value isn't given, the top of the stack will be returned.
<ww:push />	Pushes a value onto the Value Stack.
<ww:param />	Sets a parent tag's parameter. This tag is used

	only inside another tag to set the value of some property of the parent tag.
<ww:set />	Sets the value of an object in the Value Stack to a scope (page, stack, application, session). If the value is not given, the top of the stack is used. If the scope is not given, the default scope of "webwork" is used.
<ww:url />	Builds an encoded URL.

```
EXAMPLE NEEDED.
```

Componentisation tags

<ww:action />	Executes an Action from within the context of a taglib. The body of the tag is used to display the results of the action invocation.
<ww:bean />	Creates a JavaBean, instantiate its properties and place it in the ActionContext for later use.
<ww:include />	Includes another page or action.

```
EXAMPLE NEEDED.
```

Flow control tags

This if-else set of tags works just like if-else scriptlets.

<ww:if />	Conditional execution path. That is, evaluates the tag body if a boolean expression is true.
<ww:else />	Negative execution path for the if tag. That is, if the preceeding conditional tag's boolean expression evaluated to false, then evaluate this tag's body.
<ww:elseif />	Negative conditional execution path for the if tag. That is, if the preceeding conditional tag's boolean expression evaluated to false and if this tag's boolean expression evaluates to true, then evaluate this tag's body.

```
EXAMPLE NEEDED.
```

Iteration tags

<ww:iterator />	Iterates over a collection.
<ww:generator />	Generates iterators.
<ww:append />	Appends several iterators.
<ww:subset />	Gets a subset of an iterator.
<ww:merge />	Merges several iterators into one.
<ww:sort />	Sorts an iterator.

```
EXAMPLE NEEDED.
```

UI tags

The UI tags wrap generic HTML controls while providing tight integration with the core framework. The tags have been designed to minimize the amount of logic in compiled code and delegate the actual rendering of HTML to a template system. The UI tags attempt to cover the most common scenarios, while providing a Component Tag for creating custom components. The UI tags also provide built-in support for displaying inline error messages.

There is a separate lesson about WebWork UI Tags which explains in detail how they work, how you could cusomize their appearance through the use of templates, how to create custom components, etc.

Go to WebWork UI Tags Lesson.

VUI(Voice UI) tags

`<ww:audio />`	???
`<ww:prompt />`	???
`<ww:filled />`	???
`<ww:log />`	???

Volunteers needed to write this part.

Internationalisation tags

`<ww:text />`	Prints out an internationalized string.
`<ww:i18n />`	Places a resource bundle on the Value Stack, for access by the text tag.

Previous Lesson | Next Lesson

This page last changed on Mar 30, 2006 by scud.

Lesson 4.1.1: WebWork UI Tags

In WebWork, the UI tags wrap generic HTML controls while providing tight integration with the core framework. The tags have been designed to minimize the amount of logic in compiled code and delegate the actual rendering of HTML to a template system. The UI tags attempt to cover the most common scenarios, while providing a Component Tag for creating custom components. The UI tags also provide built-in support for displaying inline error messages.

This lesson tries to explain how to take advantage of the UI tags to build forms and other graphical controls and, by explaining how the template system works, teaches you how to change the look of existing components and create your own UI components.

Building forms:

WebWork comes with ready-to-use tags to construct forms. Some of these tags relate directly to HTML tags that are used to make forms and you probably can figure them out by their names: `<ww:checkbox />`, `<ww:file />`, `<ww:form />`, `<ww:hidden />`, `<ww:label />`, `<ww:password />`, `<ww:radio />`, `<ww:select />`, `<ww:submit />`, `<ww:textarea />` and `<ww:textfield />`.

To build forms with these tags, place them in your page as you would do with the HTML tags. The only difference is that the parameters should be enclosed in double quotes and single quotes (`key="'value'"`). That's because names that are not single-quoted are evaluated against the Value Stack.

Let's check out an example:

ex01-index.jsp:

```
<%@ taglib uri="webwork" prefix="ww" %>
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1.1 - Example 1</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<style type="text/css">
        .errorMessage { color: red; }
</style>
</head>

<body>

<p>UI Form Tags Example:</p>

<ww:form action="'formProcessing.action'" method="'post'">
        <ww:checkbox name="'checkbox'" label="'A checkbox'" fieldValue="'checkbox_value'" />
        <ww:file name="'file'" label="'A file field'" />
        <ww:hidden name="'hidden'" value="'hidden_value'" />
        <ww:label label="'A label'" />
        <ww:password name="'password'" label="'A password field'" />
        <ww:radio name="'radio'" label="'Radio buttons'" list="{'One', 'Two', 'Three'}" />
        <ww:select name="'select'" label="'A select list'" list="{'One', 'Two', 'Three'}"
                emptyOption="true" />
        <ww:textarea name="'textarea'" label="'A text area'" rows="'3'" cols="'40'" />
        <ww:textfield name="'textfield'" label="'A text field'" />
        <ww:submit value="'Send Form'" />
</ww:form>
```

```
</body>
</html>
```

HTML result after processing ex01-index.jsp:

```
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1.1 - Example 1</title>
<style type="text/css">
  .errorMessage { color: red; }
</style>
</head>

<body>

<p>UI Form Tags Example:</p>

<table>
<form
action="formProcessing.action" method="post" >




<tr>
<td valign="top" colspan="2">

<table width="100%" border="0" cellpadding="0" cellspacing="0">
<tr><td valign="top">
<input type="checkbox"
name="checkbox"
value="checkbox_value"
/>
</td>
<td width="100%" valign="top">
<span class="checkboxLabel">
A checkbox
</span>
</td>
</tr>
</table>
</td>
</tr>





<tr>
<td align="right" valign="top">

<span class="label">

A file field:
</span>
</td>

<td>

<input type="file"
name="file"
/>

</td>
</tr>

  <input
type="hidden"
name="hidden" value="hidden_value" />
```

```
<tr>
<td align="right" valign="top">

<span class="label">

A label:
</span>
</td>

<td>
<label> </label>
</td>
</tr>






<tr>
<td align="right" valign="top">

<span class="label">

A password field:
</span>
</td>

<td>

<input type="password"
name="password"


/>

</td>
</tr>






<tr>
<td align="right" valign="top">

<span class="label">

Radio buttons:
</span>
</td>

<td>





<input
type="radio"
name="radio"
id="radioOne"
value="One" />
<label for="radioOne">One</label>






<input
type="radio"
name="radio"
id="radioTwo"
value="Two" />
<label for="radioTwo">Two</label>
```

```
<input
type="radio"
name="radio"
id="radioThree"
value="Three" />
<label for="radioThree">Three</label>


</td>
</tr>






<tr>
<td align="right" valign="top">

<span class="label">

A select list:
</span>
</td>

<td>

<select name="select"
>


<option value=""></option>





<option value="One"
>One</option>





<option value="Two"
>Two</option>





<option value="Three"
>Three</option>


</select>

</td>
</tr>






<tr>
<td align="right" valign="top">

<span class="label">

A text area:
</span>
</td>

<td>

<textarea name="textarea"
```

```
cols="40"
rows="3"
></textarea>

</td>
</tr>






<tr>
<td align="right" valign="top">

<span class="label">

A text field:
</span>
</td>

<td>

<input type="text"
name="textfield"
/>

</td>
</tr>

  <tr>
<td colspan="2"><div
align="right" ><input
type="submit"
value="Send Form" /></div>
</td>
</tr>

</form>
</table>


</body>
</html>
```

xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
                <action name="formProcessing" class="lesson04_01_01.FormProcessingAction">
                        <result name="input" type="dispatcher">ex01-index.jsp</result>
                        <result name="success" type="dispatcher">ex01-success.jsp</result>
                        <interceptor-ref name="validationWorkflowStack" />
                </action>
        </package>
</xwork>
```

FormProcessingAction.java:

```
package lesson04_01_01;

import com.opensymphony.xwork.ActionSupport;
```

```
public class FormProcessingAction extends ActionSupport {
        private String checkbox;
        private String file;
        private String hidden;
        private String password;
        private String radio;
        private String select;
        private String textarea;
        private String textfield;

        public String getCheckbox() { return checkbox; }
        public String getFile() { return file; }
        public String getHidden() { return hidden; }
        public String getPassword() { return password; }
        public String getRadio() { return radio; }
        public String getSelect() { return select; }
        public String getTextarea() { return textarea; }
        public String getTextfield() { return textfield; }

        public void setCheckbox(String checkbox) { this.checkbox = checkbox; }
        public void setFile(String file) { this.file = file; }
        public void setHidden(String hidden) { this.hidden = hidden; }
        public void setPassword(String password) { this.password = password; }
        public void setRadio(String radio) { this.radio = radio; }
        public void setSelect(String select) { this.select = select; }
        public void setTextarea(String textarea) { this.textarea = textarea; }
        public void setTextfield(String textfield) { this.textfield = textfield; }

        public String execute() throws Exception {
                return SUCCESS;
        }
}
```

FormProcessingAction-validation.xml:

```
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-validator-1.0.dtd">

<validators>
  <field name="checkbox">
    <field-validator type="requiredstring">
      <message>Please, check the checkbox.</message>
    </field-validator>
  </field>

  <field name="file">
    <field-validator type="requiredstring">
      <message>Please select a file.</message>
    </field-validator>
  </field>

  <field name="password">
    <field-validator type="requiredstring">
      <message>Please type something in the password field.</message>
    </field-validator>
  </field>

  <field name="radio">
    <field-validator type="requiredstring">
      <message>Please select a radio button.</message>
    </field-validator>
  </field>

  <field name="select">
    <field-validator type="requiredstring">
      <message>Please select an option from the list.</message>
    </field-validator>
  </field>

  <field name="textarea">
    <field-validator type="requiredstring">
      <message>Please type something in the text area.</message>
    </field-validator>
  </field>

  <field name="textfield">
    <field-validator type="requiredstring">
      <message>Please type something in the text field.</message>
    </field-validator>
  </field>
</validators>
```

ex01-success.jsp:

```
<%@ taglib uri="webwork" prefix="ww" %>
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1.1 - Example 1</title>
</head>

<body>

<p>UI Form Tags Example result:</p>

<ul>
  <li>checkbox: <ww:property value="checkbox" /></li>
  <li>file: <ww:property value="file" /></li>
  <li>hidden: <ww:property value="hidden" /></li>
  <li>password: <ww:property value="password" /></li>
  <li>radio: <ww:property value="radio" /></li>
  <li>select: <ww:property value="select" /></li>
  <li>textarea: <ww:property value="textarea" /></li>
  <li>textfield: <ww:property value="textfield" /></li>
</ul>

</body>
</html>
```

Notice how much cleaner `ex01-index.jsp` is, compared to its HTML result. The default layout of the form components is a table layout, with the label on the left column and the field to the right. You can learn how to create your own layouts when we explain the template system, below.

Another thing to notice is the reference to the `validationWorkflowStack` in the action's configuration. This makes WebWork validate the parameters that are sent to our actions according to a configuration file we place in the same location as the action class - in our case, `FormProcessingAction-validation.xml` (see Validation). In case something is not valid, it prevents the action from executing and dispatches the request to the input result with error messages attached to each field (using the method `addFieldError(String fieldName, String errorMessage)`).

But don't worry about how the validation framework works for now. Run the example and try leaving some fields blank. You will see that the UI tags provide error messages that integrate with the validation framework and that's what we want do demonstrate here. This separation of concerns can help programmers and designers concentrate more on their part of the work.

Try the example!

Other UI Controls:

Besides the standard form controls that HTML designers are already familiar with, WebWork provides some other controls and also the ability to create a custom control. Let's take a look at the custom controls that are already provided by WebWork:

<ww:checkboxlist />	Works just like the `<ww:radio />` tag, but with check boxes instead of radio buttons. It gets the keys and values from a collection and creates a list of checkboxes, all with the same name.
<ww:combobox />	Simulates a combo box, which is a control that mixes a selection list with a text field. It does this by placing a text field with a `<select />` list right below it and a JavaScript code that fills the text field with the selection of the list every time it changes.
<ww:tabbedpane />	Help needed here.
<ww:token />	Help needed here.

The Template System:

WebWork uses the Velocity template system to render the actual HTML output for all UI tags. A default implementation of all templates has been included with the core distribution allowing users to use WebWork's UI tags "out of the box". Templates can be edited individually or replaced entirely allowing for complete customization of the resulting HTML output. In addition, the default template can be overridden on a per tag basis allowing for a very fine level of control. The default templates are located in the `webwork-2.1.1.jar` file under `/template/xhtml`.

If you unpack `webwork-2.1.1.jar` and look under the `/template/xhtml` directory you will see a bunch of velocity templates. Most of them correspond to a specific UI Tag, and those have the name of the tag they render. If you're familiar with Velocity, I recommend you analyse the template files to see what you're capable of doing with them. Since version 2.1, there's also a `/template/simple` directory, which is a simpler version of the HTML form controls (just the control, no table or label).

If you want do display your UI components in a different layout than the one that comes with WebWork, you can:

- Edit and replace the files in `/template/xhtml` (repack the JAR or create the same directory structure somewhere else and make sure your container looks that path before the JAR);

- Change the location of the templates by editing the `webwork.ui.theme` property in `webwork.properties` (file that should be placed in the root of your classpath);

- Specifying the location of the templates for each tag individually using the theme or the template property. The former allows you to specify the directory where all templates are (thus, WebWork looks for templates with the same name as the ones in `/template/xhtml`), while the latter allows you to indicate the exact template to be used for that component.

Read more: Themes and Templates

The third approach is demonstrated in the example below. Note that, by default, the specified theme directory should be under `/template` and the specified template file should be under `/template/xhtml`.

ex02.jsp:

```
<%@ taglib uri="webwork" prefix="ww" %>
```

```
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1.1 - Example 2</title>
</head>

<body>

<p>Template Change Example:</p>

<p><ww:checkbox name="'checkbox'" label="'A checkbox'" fieldValue="'checkbox_value'"
theme="'mytheme'" /></p>

<p><ww:textfield name="'textfield'" label="'A text field'" template="mytextfield.vm" /></p>

</body>
</html>
```

/template/mytheme/checkbox.vm:

```
<div align="center">
        <input type="checkbox"
                name="$!webwork.htmlEncode($parameters.name)"
                value="$!webwork.htmlEncode($parameters.fieldValue)"
        #if ($parameters.nameValue) checked="checked" #end
        #if ($parameters.disabled == true) disabled="disabled" #end
        #if ($parameters.tabindex) tabindex="$!webwork.htmlEncode($parameters.tabindex)" #end
        #if ($parameters.onchange) onchange="$!webwork.htmlEncode($parameters.onchange)" #end
        #if ($parameters.id) id="$!webwork.htmlEncode($parameters.id)" #end
        /><br />
        $!webwork.htmlEncode($parameters.label)
</div>
```

/template/xhtml/mytextfield.vm:

```
<div align="center">
        <input type="text"
                name="$!webwork.htmlEncode($parameters.name)"
        #if ($parameters.size) size="$!webwork.htmlEncode($parameters.size)" #end
        #if ($parameters.maxlength) maxlength="$!webwork.htmlEncode($parameters.maxlength)"
#end
        #if ($parameters.nameValue) value="$!webwork.htmlEncode($parameters.nameValue)" #end
        #if ($parameters.disabled == true) disabled="disabled" #end
        #if ($parameters.readonly) readonly="readonly" #end
        #if ($parameters.onkeyup) onkeyup="$!webwork.htmlEncode($parameters.onkeyup)" #end
        #if ($parameters.tabindex) tabindex="$!webwork.htmlEncode($parameters.tabindex)" #end
        #if ($parameters.onchange) onchange="$!webwork.htmlEncode($parameters.onchange)" #end
        #if ($parameters.id) id="$!webwork.htmlEncode($parameters.id)" #end
        /><br />
        $!webwork.htmlEncode($parameters.label)
</div>
```

HTML result after processing ex02.jsp:

```
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1.1 - Example 2</title>
</head>

<body>

<p>Template Change Example:</p>

<p><div align="center">
  <input type="checkbox"
```

```
          name="checkbox"
          value="checkbox_value"
             /><br />
   A checkbox
 </div></p>

 <p><div align="center">
   <input type="text"
                                            name="textfield"
                    /><br />
   A text field
 </div></p>

 </body>
 </html>
```

Try the example!

Building Customized UI Components:

There are some situations in which none of the UI Components that come bundled with WebWork fit your requirements. In this case, the recommended approach would be to create your own custom component. In this way, you keep your web page clean of layout and error-checking issues and also promote component reuse.

To create a custom component, just create a Velocity template for it, just like the ones that already exist. To place it in a web page, use the `<ww:component />` tag and specify the location of the template in its `template` parameter.

To pass parameters to be used by your template, use the `<ww:param />` tag (see lesson 4.1). The example below demonstrates the creation of a custom date field.

ex03.jsp:

```
<%@ taglib uri="webwork" prefix="ww" %>
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1.1 - Example 3</title>
</head>

<body>
<p>Custom Component Example:</p>

<p>
<ww:component template="datefield.vm">
        <ww:param name="'label'" value="'Date'" />
        <ww:param name="'name'" value="'mydatefield'" />
        <ww:param name="'size'" value="3" />
</ww:component>
</p>

</body>
</html>
```

/template/xhtml/datefield.vm:

```
#set ($name = $parameters.get('name'))
#set ($size = $parameters.get('size'))
```

```
#set ($yearSize = $size * 2)

$parameters.get('label'):
<input type="text" name="${name}.day" size="$size" /> /
<input type="text" name="${name}.month" size="$size" /> /
<input type="text" name="${name}.year" size="$yearSize" /> (dd/mm/yyyy)
```

HTML result after processing ex03.jsp:

```
<html>
<head>
<title>WebWork Tutorial - Lesson 4.1.1 - Example 3</title>
</head>
<body>
<p>Custom Component Example:</p>

<p>
Date:
<input type="text" name="mydatefield.day" size="3" /> /
<input type="text" name="mydatefield.month" size="3" /> /
<input type="text" name="mydatefield.year" size="6" /> (dd/mm/yyyy)
</p>

</body>
</html>
```

Try the example!

Previous Lesson | Next Lesson

This page last changed on Mar 30, 2006 by scud.

Lesson 4.2: Using Velocity with WebWork

There are two ways of using Velocity as the view.

- Using the `velocity` result-type to render velocity templates;
- Registering `WebWorkVelocityServlet` in your `web.xml` file to render Velocity templates accessed directly through browser requests.

To use the second approach, we have to modify `web.xml` and add a servlet and a servlet mapping for `WebWorkVelocityServlet`, as demonstrated below:

```
<servlet>
        <servlet-name>velocity</servlet-name>
        <servlet-class>com.opensymphony.webwork.views.velocity.WebWorkVelocityServlet</servlet-class>
        <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
        <servlet-name>velocity</servlet-name>
        <url-pattern>*.vm</url-pattern>
</servlet-mapping>
```

Read more: xwork.xml

Using `velocity` result-type means that Velocity templates can only be rendered through an action, i.e., request to `.vm` pages will not render the file and it will be returned as plain text. If you choose this approach, it's recommended that you place your Velocity files under `WEB-INF` so they become unaccessible.

Using `WebWorkVelocityServlet` means that Velocity templates can be rendered through requests to `.vm` pages. That also means that you should implement security checks in your templates so an user doesn't access it directly witout going through an action first (if that is required).

No matter which approach you choose (and you can choose to use both at the same time), not only all the features from Velocity are available to you when you're writing templates, but also some other functionalities, specific of WebWork, are available. It is supposed that you are already familiar with Velocity, so we will focus only in the WebWork-specific features. If that's not the case, please get started with Velocity before continuing.

The main feature of it is to provide easy access to objects that are on the Value Stack, which contains some things that WebWork provides to you automatically, because you may find them useful at some point. These are some of the things that are available in the value stack:

- The current `HttpServletRequest`;
- The current `HttpServletResponse`;
- The current `OgnlValueStack`;
- An instance of `OgnlTool`;
- All the properties of the current action class.

To access the objects in the value stack, all you have to do is use appropriate Velocity references:

- **$req** = HttpServletRequest;
- **$res** = HttpServletResponse;
- **$stack** = OgnlValueStack;
- **$ognl** = OgnlTool;
- **$name-of-property** = property of the current action class.

The example below does the same thing as the Hello example from lesson 3, but now, using a Velocity template as the result. Notice that the `<property value="person" />` tag was replaced by the `$person` reference, which returns the same thing: a property from the action class. In this example we chose to use the *`velocity`* result-type approach.

xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
	<!-- Include webwork defaults (from WebWork JAR). -->
	<include file="webwork-default.xml" />

	<!-- Configuration for the default package. -->
	<package name="default" extends="webwork-default">
		<!-- Default interceptor stack. -->
		<default-interceptor-ref name="defaultStack" />

		<!-- Action: Lesson 4.2: HelloAction using Velocity as result. -->
		<action name="helloVelocity" class="lesson03.HelloAction">
			<result name="error" type="dispatcher">ex01-index.jsp</result>
			<result name="success" type="velocity">ex01-success.vm</result>
		</action>
	</package>
</xwork>
```

HelloAction.java (same as lesson 3):

```
package lesson03;

import com.opensymphony.xwork.ActionSupport;

public class HelloAction extends ActionSupport {
	String person;
	public String getPerson() {
		return person;
	}
	public void setPerson(String person) {
		this.person = person;
	}
	public String execute() throws Exception {
		if ((person == null) || (person.length() == 0)) return ERROR;
		else return SUCCESS;
	}
}
```

ex01-index.jsp (same as lesson 3):

```
<html>
<head>
<title>WebWork Tutorial - Lesson 3 - Example 2</title>
</head>

<body>
```

```
<p>What's your name?</p>

<form action="helloVelocity.action" method="post">
<p><input type="text" name="person" /><input type="submit" /></p>
</form>

</body>
</html>
```

ex01-success.vm:

```

<html>
<head>
<title>WebWork Tutorial - Lesson 4.2 - Example 1</title>
</head>
<body>

Hello, $person

</body>
</html>
```

Try the example!

Using WebWork Tags from Velocity:

As you already know, when you switch from JSP to Velocity you lose the ability of using JSP Tags. But WebWork's Velocity Servlet provides a way of doing this through the use of #tag, #bodytag and #param velocimacros. The general syntax is:

```

#tag (name-of-tag list-of-attributes)
```

- or -

```

#bodytag (name-of-tag list-of-attributes)
        #param (key value)
        #param (key value)
...
#end
```

Let's revisit lesson 4.1.1's form example to demonstrate the usage of the UI tags from velocity:

xwork.xml:

```

<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
                <!-- Default interceptor stack. -->
                <default-interceptor-ref name="defaultStack" />

                <!-- Actions: Lesson 4.2: FormProcessingAction using Velocity. -->
                <action name="formProcessingVelocityIndex"
class="lesson04_02.FormProcessingIndexAction">
                        <result name="success" type="velocity">ex02-index.vm</result>
                </action>
                <action name="formProcessingVelocity"
class="lesson04_01_01.FormProcessingAction">
                        <result name="input" type="velocity">ex02-index.vm</result>
                        <result name="success" type="velocity">ex02-success.vm</result>
                        <interceptor-ref name="validationWorkflowStack" />
                </action>
        </package>
</xwork>
```

ex02-index.vm:

```
<html>
<head>
<title>WebWork Tutorial - Lesson 4.2 - Example 2</title>
<style type="text/css">
  .errorMessage { color: red; }
</style>
</head>

<body>

<p>UI Form Tags Example using Velocity:</p>

#bodytag (Form "action='formProcessingVelocity.action'" "method='post'")
        #tag (Checkbox "name='checkbox'" "label='A checkbox'" "fieldValue='checkbox_value'")
        #tag (File "name='file'" "label='A file field'")
        #tag (Hidden "name='hidden'" "value='hidden_value'")
        #tag (Label "label='A label'")
        #tag (Password "name='password'" "label='A password field'")
        #tag (Radio "name='radio'" "label='Radio buttons'" "list={'One', 'Two', 'Three'}")
        #tag (Select "name='select'" "label='A select list'" "list={'One', 'Two', 'Three'}"
                "emptyOption=true")
        #tag (Textarea "name='textarea'" "label='A text area'" "rows='3'" "cols='40'")
        #tag (TextField "name='textfield'" "label='A text field'")
        #tag (Submit "value='Send Form'")
#end

</body>
</html>
```

ex02-success.vm:

```
<html>
<head>
<title>WebWork Tutorial Lesson 4.2 - Example 2</title>
</head>

<body>

<p>UI Form Tags Example result using Velocity:</p>

<ul>
        <li>checkbox: $!checkbox</li>
        <li>file: $!file</li>
        <li>hidden: $!hidden</li>
        <li>password: $!password</li>
        <li>radio: $!radio</li>
        <li>select: $!select</li>
        <li>textarea: $!textarea</li>
        <li>textfield: $!textfield</li>
</ul>
```

```
</body>
</html>
```

FormProcessingAction.java (same as lesson 4.1.1):

```
package lesson04_01_01;

import com.opensymphony.xwork.ActionSupport;

public class FormProcessingAction extends ActionSupport {
        private String checkbox;
        private String file;
        private String hidden;
        private String password;
        private String radio;
        private String select;
        private String textarea;
        private String textfield;

        public String getCheckbox() { return checkbox; }
        public String getFile() { return file; }
        public String getHidden() { return hidden; }
        public String getPassword() { return password; }
        public String getRadio() { return radio; }
        public String getSelect() { return select; }
        public String getTextarea() { return textarea; }
        public String getTextfield() { return textfield; }

        public void setCheckbox(String checkbox) { this.checkbox = checkbox; }
        public void setFile(String file) { this.file = file; }
        public void setHidden(String hidden) { this.hidden = hidden; }
        public void setPassword(String password) { this.password = password; }
        public void setRadio(String radio) { this.radio = radio; }
        public void setSelect(String select) { this.select = select; }
        public void setTextarea(String textarea) { this.textarea = textarea; }
        public void setTextfield(String textfield) { this.textfield = textfield; }

        public String execute() throws Exception {
                return SUCCESS;
        }
}
```

FormProcessingAction-validation.xml (same as lesson 4.1.1):

```
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator
1.0//EN" "http://www.opensymphony.com/xwork/xwork-validator-1.0.dtd">

<validators>
  <field name="checkbox">
    <field-validator type="requiredstring">
      <message>Please, check the checkbox.</message>
    </field-validator>
  </field>

  <field name="file">
    <field-validator type="requiredstring">
      <message>Please select a file.</message>
    </field-validator>
  </field>

  <field name="password">
    <field-validator type="requiredstring">
      <message>Please type something in the password field.</message>
    </field-validator>
  </field>

  <field name="radio">
    <field-validator type="requiredstring">
      <message>Please select a radio button.</message>
    </field-validator>
  </field>

  <field name="select">
    <field-validator type="requiredstring">
      <message>Please select an option from the list.</message>
    </field-validator>
  </field>

  <field name="textarea">
    <field-validator type="requiredstring">
      <message>Please type something in the text area.</message>
    </field-validator>
  </field>

  <field name="textfield">
    <field-validator type="requiredstring">
      <message>Please type something in the text field.</message>
    </field-validator>
  </field>
</validators>
```

Try the example!

—

The example above does not use the `#param` tag. So, let's revisit another example from lesson 4.1.1 - custom components:

ex03.vm:

```
<html>
<head>
<title>WebWork Tutorial - Lesson 4.2 - Example 3</title>
</head>

<body>

<p>Custom Component Example:</p>

<p>
#bodytag (Component "template=datefield.vm")
        #param ("label" "Date")
        #param ("name" "mydatefield")
        #param ("size" "3")
#end
</p>

</body>
</html>
```

/template/xhtml/datefield.vm (same as lesson 4.1.1):

```
#set ($name = $parameters.get('name'))
#set ($size = $parameters.get('size'))
#set ($yearSize = $size * 2)

$parameters.get('label'):
<input type="text" name="${name}.day" size="$size" /> /
<input type="text" name="${name}.month" size="$size" /> /
<input type="text" name="${name}.year" size="$yearSize" /> (dd/mm/yyyy)
```

Notice that, this time, we did not enclose `Date` and `mydatefield` with single quotes, as we had to do when we used the JSP tag.

Try the example!

Previous Lesson | Next Lesson

This page last changed on Mar 30, 2006 by scud.

Lesson 4.3: Using Freemarker with WebWork

Freemarker is a powerfull template engine that competes with Velocity. You can learn more about it in the project's homepage: http://freemarker.sourceforge.net.

First of all, to use Freemarker with Webwork, you have to place the freemarker.jar in your `WEB-INF\lib` folder. You can download the distribution here.

After that, just configure `web.xml` and start writing your templates, as explained below.

web.xml:

To use Freemarker as the view, you need to modify `web.xml` and add a servlet and a servlet mapping for `FreemarkerServlet`, as demonstrated below:

```
<servlet>
        <servlet-name>freemarker</servlet-name>
        <servlet-class>com.opensymphony.webwork.views.freemarker.FreemarkerServlet</servlet-class>
                <!-- FreemarkerServlet settings: -->
                <init-param>
                        <param-name>TemplatePath</param-name>
                        <param-value>/</param-value>
                </init-param>
                <init-param>
                        <param-name>NoCache</param-name>
                        <param-value>true</param-value>
                </init-param>
                <init-param>
                        <param-name>ContentType</param-name>
                        <param-value>text/html</param-value>
                </init-param>
                <init-param>
                        <param-name>default_encoding</param-name>
                        <param-value>ISO-8859-1</param-value>
                </init-param>
                <init-param>
                        <param-name>number_format</param-name>
                        <param-value>0.##########</param-value>
                </init-param>
        <load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
        <servlet-name>freemarker</servlet-name>
        <url-pattern>*.ftl</url-pattern>
</servlet-mapping>
```

The configuration above means that Freemarker templates can be rendered through requests to `.ftl` pages. That also means that you should implement security checks in your templates so an user doesn't access it directly without going through an action first (if that is required). But you can always place your Freemarker files under WEB-INF so they become unaccessible to direct requests. We will use the latter approach in our examples.

Inside a Freemarker template, you will have access to every object managed by WebWork with the following syntax:

- **$stack** = OgnlValueStack;
- **$webwork** = FreemarkerWebWorkUtil, a toolbox providing services like formatting url, accessing the value stack, etc;
- **$name-of-property** = property retrieved from the value stack. If that fails, it looks up an attribute with that name in the HttpServletRequest, HttpSession and ServletContext, in that order;
- **$Request** = HttpServletRequest;
- **$Session** = HttpServletResponse;
- **$Application** = OgnlValueStack.

The example below does the same thing as example 2 from lesson 3, but now, using Freemarker templates.

xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
                <!-- Default interceptor stack. -->
                <default-interceptor-ref name="defaultStack" />

                <!-- Action: Lesson 4.3: HelloAction. -->
                <action name="indexFreemarker" class="com.opensymphony.xwork.ActionSupport">
                        <result name="success"
type="dispatcher">/WEB-INF/ftl/lesson3/index.ftl</result>
                </action>

                <action name="helloFreemarker" class="lesson03.HelloAction">
                        <result name="error"
type="dispatcher">/WEB-INF/ftl/lesson3/index.ftl</result>
                        <result name="success"
type="dispatcher">/WEB-INF/ftl/lesson3/success.ftl</result>
                </action>
        </package>
</xwork>
```

HelloAction.java (same as lesson 3):

```
package lesson03;

import com.opensymphony.xwork.ActionSupport;

public class HelloAction extends ActionSupport {
        String person;
        public String getPerson() {
                return person;
        }
        public void setPerson(String person) {
                this.person = person;
        }
        public String execute() throws Exception {
                if ((person == null) || (person.length() == 0)) return ERROR;
                else return SUCCESS;
        }
}
```

ex02-index.ftl

```
<#assign ww=JspTaglibs["/WEB-INF/lib/webwork.tld"] />

<html>
<head>
<title>WebWork Tutorial - Lesson 4.3 - Example 1</title>
</head>

<body>

<p>Click <a href="${wwUtil.buildUrl('indexFreemarker.action')}">here</a> to reload this
page.</p>

<@ww.form name="'nameForm'" action="'helloFreemarker.action'" method="'POST'">
        <@ww.textfield label="'What is your name ?'" name="'person'" value="person" size="20"/>
        <@ww.submit name="'submit'" value="'Submit'"/>
</@ww.form>

</body>
</html>
```

If you don't want to use WebWork's UI Tags, you could do it like this:

ex02-index-notags.ftl

```
<html>
<head>
<title>WebWork Tutorial - Lesson 4.3 - Example 1</title>
</head>

<body>

<p>Click <a href="${wwUtil.buildUrl('indexFreemarker.action')}">here</a> to reload this
page.</p>

<form name="nameForm" action="${wwUtil.buildUrl('helloFreemarker.action')}" method="POST">
        What is your name ?
        <input type="text" name="person" value="${person}" size="20">
        <input type="submit" name="submit" value="Submit">
</form>
</body>
</html>
```

However, if you choose no to use tags, it's recommended that you use Freemarker Macros to write the form elements.

ex02-success.ftl:

```
<#assign ww=JspTaglibs["/WEB-INF/lib/webwork.tld"] />

<html>
<head>
<title>WebWork Tutorial - Lesson 4.3 - Example 1</title>
</head>
<body>

Come from the property WW tag (taglibs support) : <@ww.property value="person"/> <br>
Come from the Freemarker lookup in the WW stack : ${person}

</body>
</html>
```

You can use either WebWork `property` tag or the Freemarker `$person` reference. Both of them return the same thing: a property from the action class.

Previous Lesson | Next Lesson

TutorialLesson06

This page last changed on Mar 30, 2006 by scud.

Lesson 5: Interceptors

Interceptors allow arbitrary code to be included in the call stack for your action before and/or after processing the action, which can vastly simplify your code itself and provide excellent opportunities for code reuse. Many of the features of XWork and WebWork are implemented as interceptors and can be applied via external configuration along with your own Interceptors in whatever order you specify for any set of actions you define.

In other words, when you access a `*.action` URL, WebWork's `ServletDispatcher` proceeds to the invocation of the an action object. Before it is executed, however, the invocation can be intercepted by another object, that is hence called interceptor. To have an interceptor executed before (or after) a given action, just configure `xwork.xml` properly, like the example below, taken from lesson 4.1.1:

Interceptor configuration from lesson 4.1.1:

```
<action name="formProcessing" class="lesson04_01_01.FormProcessingAction">
        <result name="input" type="dispatcher">ex01-index.jsp</result>
        <result name="success" type="dispatcher">ex01-success.jsp</result>
        <interceptor-ref name="validationWorkflowStack" />
</action>
```

As you can see, lesson 4.1.1's `formProcessing` Action uses the `validationWorkflowStack`. That is an interceptor stack, which organizes a bunch of interceptors in the order in which they are to be executed. That stack is configured in `webwork-default.xml`, so all we have to do to use it is declare a `<interceptor-ref />` under the action configuration or a `<default-interceptor-ref />`, under package configuration, as seen in lesson 3's first example:

Interceptor configuration from lesson 3.1:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
                <!-- Default interceptor stack. -->
                <default-interceptor-ref name="defaultStack" />

                <!-- Action: Lesson 03: HelloWebWorldAction. -->
                <action name="helloWebWorld" class="lesson03.HelloWebWorldAction">
                        <result name="success" type="dispatcher">ex01-success.jsp</result>
                </action>
        </package>
</xwork>
```

But let's see how it works from scracth:

1. Create an interceptor class, which is a class that implements the `com.opensymphony.xwork.interceptor.Interceptor` interface (bundled in `xwork-1.0.jar`);

2. Declare the class in your XML configuration file (`xwork.xml`) using the element `<interceptor />` nested within `<interceptors />`;
3. Create stacks of interceptors, using the `<interceptor-stack />` element (optional);
4. Determine which interceptors are used by which action, using `<interceptor-ref />` or `<default-interceptor-ref />`. The former defines the interceptors to be used in a specific action, while the latter determines the default interceptor stack to be used by all actions that do not specify their own `<interceptor-ref />`.

Looking inside `webwork-default.xml` we can see how it's done:

webwork-default.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <package name="webwork-default">
                <result-types>
                        <result-type name="dispatcher" default="true"
                                class="com.opensymphony.webwork.dispatcher.ServletDispatcherResult"/>
                        <result-type name="redirect"
                                class="com.opensymphony.webwork.dispatcher.ServletRedirectResult"/>
                        <result-type name="velocity"
                                class="com.opensymphony.webwork.dispatcher.VelocityResult"/>
                        <result-type name="chain"
                                class="com.opensymphony.xwork.ActionChainResult"/>
                        <result-type name="xslt"
                                class="com.opensymphony.webwork.views.xslt.XSLTResult"/>
                </result-types>

                <interceptors>
                        <interceptor name="timer"
                                class="com.opensymphony.xwork.interceptor.TimerInterceptor"/>
                        <interceptor name="logger"
                                class="com.opensymphony.xwork.interceptor.LoggingInterceptor"/>
                        <interceptor name="chain"
                                class="com.opensymphony.xwork.interceptor.ChainingInterceptor"/>
                        <interceptor name="static-params"
                                class="com.opensymphony.xwork.interceptor.StaticParametersInterceptor"/>
                        <interceptor name="params"
                                class="com.opensymphony.xwork.interceptor.ParametersInterceptor"/>
                        <interceptor name="model-driven"
                                class="com.opensymphony.xwork.interceptor.ModelDrivenInterceptor"/>
                        <interceptor name="component"
                                class="com.opensymphony.xwork.interceptor.component.ComponentInterceptor"/>
                        <interceptor name="token"
                                class="com.opensymphony.webwork.interceptor.TokenInterceptor"/>
                        <interceptor name="token-session"
                                class="com.opensymphony.webwork.interceptor.TokenSessionStoreInterceptor"/>
                        <interceptor name="validation"
                                class="com.opensymphony.xwork.validator.ValidationInterceptor"/>
                        <interceptor name="workflow"
                                class="com.opensymphony.xwork.interceptor.DefaultWorkflowInterceptor"/>
                        <interceptor name="servlet-config"
                                class="com.opensymphony.webwork.interceptor.ServletConfigInterceptor"/>
                        <interceptor name="prepare"
                                class="com.opensymphony.xwork.interceptor.PrepareInterceptor"/>
                        <interceptor name="conversionError"
                                class="com.opensymphony.webwork.interceptor.WebWorkConversionErrorIntercepto
                        <interceptor-stack name="defaultStack">
                                <interceptor-ref name="static-params"/>
                                <interceptor-ref name="params"/>
                                <interceptor-ref name="conversionError"/>
                        </interceptor-stack>
                        <interceptor-stack name="validationWorkflowStack">
                                <interceptor-ref name="defaultStack"/>
                                <interceptor-ref name="validation"/>
                                <interceptor-ref name="workflow"/>
                        </interceptor-stack>
                </interceptors>
        </package>
```

```
</xwork>
```

Since we included `webwork-default.xml` in our `xwork.xml`, all the interceptors and stacks above are available for us to use in our actions. Here's what these interceptors do:

- **timer**: clocks how long the action (including nested interceptors and view) takes to execute;
- **logger**: logs the action being executed;
- **chain**: makes the previous action's properties available to the current action. Used to make action chaining (reference: Result Types);
- **static-params**: sets the parameters defined in `xwork.xml` onto the action. These are the `<param />` tags that are direct children of the `<action />` tag;
- **params**: sets the request (POST and GET) parameters onto the action class. We have seen an example of this in lesson 3;
- **model-driven**: if the action implements `ModelDriven`, pushes the `getModel()` result onto the Value Stack;
- **component**: enables and makes registered components available to the actions. (reference: IoC & Components);
- **token**: checks for valid token presence in action, prevents duplicate form submission;
- **token-session**: same as above, but storing the submitted data in session when handed an invalid token;
- **validation**: performs validation using the validators defined in `{Action}-validation.xml` (reference: Validation). We've seen an example of this in lesson 4.1.1;
- **workflow**: calls the validate method in your action class. If action errors created then it returns the `INPUT` view. Good to use together with the validation interceptor (reference: Validation);
- **servlet-config**: give access to `HttpServletRequest` and `HttpServletResponse` (think twice before using this since this ties you to the Servlet API);
- **prepare**: allows you to programmatic access to your Action class before the parameters are set on it.;
- **conversionError**: help needed here.

Building your own Interceptor

If none of the above interceptors suit your particular need, you will have to implement your own interceptor. Fortunately, this is an easy task to accomplish. Suppose we need an interceptor that places a greeting in the Session according to the time of the day (morning, afternoon or evening). Here's how we could implement it:

GreetingInterceptor.java:

```
package lesson05;

import java.util.Calendar;
import com.opensymphony.xwork.interceptor.Interceptor;
import com.opensymphony.xwork.ActionInvocation;

public class GreetingInterceptor implements Interceptor {
        public void init() { }
        public void destroy() { }
        public String intercept(ActionInvocation invocation) throws Exception {
                Calendar calendar = Calendar.getInstance();
                int hour = calendar.get(Calendar.HOUR_OF_DAY);
                String greeting = (hour < 6) ? "Good evening" :
                        ((hour < 12) ? "Good morning":
                        ((hour < 18) ? "Good afternoon": "Good evening"));

                invocation.getInvocationContext().getSession().put("greeting", greeting);

                String result = invocation.invoke();

                return result;
        }
}
```

xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
                <interceptors>
                        <interceptor name="greeting"
class="section02.lesson05.GreetingInterceptor" />
                </interceptors>

                <!-- Action: Lesson 5: GreetingInterceptor. -->
                <action name="greetingAction" class="lesson05.GreetingAction">
                        <result name="success" type="velocity">ex01-result.vm</result>
                        <interceptor-ref name="greeting" />
                </action>
        </package>
</xwork>
```

GreetingAction.java:

```
package lesson05;

import com.opensymphony.xwork.ActionSupport;

public class GreetingAction extends ActionSupport {
        public String execute() throws Exception {
                return SUCCESS;
        }
}
```

ex01-result.vm:

```
<html>
<head>
<title>WebWork Tutorial - Lesson 5 - Example 1</title>
</head>
<body>

#set ($ses = $req.getSession())
<p><b>${ses.getAttribute('greeting')}!</b></p>

</body>
</html>
```

Let's take a look at our interceptor class first. As explained before, the interceptor must implement `com.opensymphony.xwork.interceptor.Interceptor`'s methods: `init()`, called during interceptor initialization, `destroy()`, called during destruction, and most importantly, `intercept(ActionInvocation invocation)`, which is where we place the code that does the work.

Notice that our interceptor returns the result from `invocation.invoke()` which is the method responsible for executing the next interceptor in the stack or, if this is the last one, the action. This means that the interceptor has the power of short-circuiting the action invocation and return a result string without executing the action at all! Use this with caution, though.

One other thing that interceptors can do is execute code after the action has executed. To do that, just place code after the invocation.invoke() call. WebWork provides an abstract class that already implements this kind of behaviour: `com.opensymphony.xwork.interceptor.AroundInterceptor`. Just extend it and implement the methods `before(ActionInvocation invocation)` and `after(ActionInvocation dispatcher, String result)`.

The `xwork.xml` configuration, the action class and the result page are pretty straightforward and require no further explanation.

Try the example!

Previous Lesson | End of Tutorial

Understanding actions

This page last changed on May 01, 2006 by dogg.

Actions

我们将创建一个你能输入名字的表单. 例如,如果你输入"Bob"并点击提交按钮,你能看到一个页面输出 "Hello, Bob!". 如果你不输入名字, 你将看到页面输出 "Hmm, you don't seem to have entered a name. Go back and try again please."

在此之前, 我们将分四步完成此任务: 创建表单, 创建Action, 注册Action, and create the landing page (or in this case, pages).

1. 创建表单

粘贴此代码到 webapp/page03.jsp:

```
<html>
<head>
        <title>A simple form with data</title>
</head>
<body>
        <p>What is your name?</p>

        <form action="form03.action" method="post">
                <p><input type="text" name="yourName"></p>
                <p><input type="submit" value="Submit your name." /></p>
        </form>

</body>
</html>
```

2. 创建Action

粘贴此代码到 src/lessons/Form03Action.java:

```
package lessons;

import com.opensymphony.xwork.ActionSupport;

public class Form03Action extends ActionSupport {

  String yourName;

  public void setYourName(String p_yourName) {
    yourName = p_yourName;
  }

  public String getYourName() {
    return yourName;
  }


  public String execute() throws Exception {
    if (yourName == null || yourName.length() == 0)
      return ERROR;
    else
```

```
        return SUCCESS;
    }

  }
```

3. 在xwork.xml中注册Action:

编辑 webapp/WEB-INF/classes/xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
  <!-- Include webwork defaults (from WebWork JAR). -->
  <include file="webwork-default.xml" />

  <!-- Configuration for the default package. -->
  <package name="default" extends="webwork-default">
    <!-- Default interceptor stack. -->
    <default-interceptor-ref name="defaultStack" />

    <!-- 02 -->
    <action name="form02" class="lessons.Form02Action">
      <result name="success" type="dispatcher">page02-success.jsp</result>
    </action>

    <!-- 03 -->
    <action name="form03" class="lessons.Form03Action">
      <result name="success" type="dispatcher">page03-success.jsp</result>
      <result name="error" type="dispatcher">page03-error.jsp</result>
    </action>

  </package>
</xwork>
```

4. 创建success 和error 页面

创建 webapp/page03-success.jsp:

```
<%@ taglib uri="webwork" prefix="ww" %>
<html>
<head>
        <title>Success page for form with data</title>
</head>
<body>

Hello, <ww:property value="yourName" />!

</body>
</html>
```

创建 webapp/page03-error.jsp:

```
<html>
<head>
        <title>Error page for form with data</title>
</head>
<body>

Hmm, you don't seem to have entered a name. Go back and try again please.

</body>
</html>
```

试试看

不要忘记编译你的Action webapp/WEB-INF/classes, 并重新启动你的应用服务器.

现在你可以尝试: 点击提交按钮看看发生了什么. 试试不输入名字会怎么样.

代码是如何工作的

此例子和上一课的例子只有两点差异.

- 当Action被调用的时候, its `setYourName()` setter method is called with the contents of the form field named `yourName`.
- After the action has been called (which is when its execute() method returns), WebWork has two options. If ERROR is returned, WebWork will return page03-error.jsp; if SUCCESS, page03-success.jsp. Just as in the last lesson, the <ww:property> tag calls the action's getter (in this case, `getYourName()`).

An html form with data, without getters or setters

For the form field named "yourName" in the previous lesson, we also had to create the getters and setters `getYourName()` and `setYourName()` in the action, as well as the private variable `yourName`. With dozens of forms and hundreds of form fields, you'll be typing thousands of getters and setters. That can get old fast. In this lesson, we'll repeat the last lesson, but without any of that extra typing.

1. 创建html表单

使用和上一课同样的JSP表单,但是要修改form标签的action属性为{{page04.action}}:

```
<html>
<head>
        <title>A simple form with data</title>
</head>
<body>
        <p>What is your name?</p>

        <form action="form04.action" method="post">
                <p><input type="text" name="yourName"></p>
                <p><input type="submit" value="Submit your name." /></p>
        </form>
```

```
</body>
</html>
```

2. 创建action

粘贴代码到{{src/lessons/Form04Action.java}}:

```
package lessons;

import com.opensymphony.xwork.ActionSupport;
import com.opensymphony.webwork.interceptor.ParameterAware;

import java.util.Map;

public class Form04Action extends ActionSupport implements ParameterAware {

  Map parameters;

  public Map getParameters() {
    return parameters;
  }

  public void setParameters(Map parameters) {
    this.parameters = parameters;
  }

  public String execute() {
    String[] yourName = (String[]) parameters.get("yourName");
    if(yourName == null || yourName[0] == null || yourName[0].length() == 0)
      return ERROR;
    else
      return SUCCESS;
  }
}
```

在 xwork.xml中注册Action:

编辑 webapp/WEB-INF/classes/xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
  <!-- Include webwork defaults (from WebWork JAR). -->
  <include file="webwork-default.xml" />

  <!-- Configuration for the default package. -->
  <package name="default" extends="webwork-default">
    <!-- Default interceptor stack. -->
    <default-interceptor-ref name="defaultStack" />

    <!-- 02 -->
    <action name="form02" class="lessons.Form02Action">
      <result name="success" type="dispatcher">page02-success.jsp</result>
    </action>

    <!-- 03 -->
    <action name="form03" class="lessons.Form03Action">
      <result name="success" type="dispatcher">page03-success.jsp</result>
```

```
        <result name="error" type="dispatcher">page03-error.jsp</result>
      </action>

      <!-- 04 -->
      <action name="form04" class="lessons.Form04Action">
        <result name="success" type="dispatcher">page04-success.jsp</result>
        <result name="error" type="dispatcher">page03-error.jsp</result>
        <interceptor-ref name="servlet-config"/>
      </action>

    </package>
  </xwork>
```

创建success 和error页

我们使用同样的error页, 但是创建一个稍微有一点不同的 success 页 page04-success.jsp. 不同之处仅仅是<ww:property>标签.

```
<%@ taglib uri="webwork" prefix="ww" %>
<html>
<head>
        <title>Success page for form with data</title>
</head>
<body>

Hello, <ww:property value="parameters.yourName" />!

</body>
</html>
```

试试看

不要忘记编译你的Action webapp/WEB-INF/classes, 并重新启动你的应用服务器.

现在,可以尝试. 装载 page04.jsp , 在文本框中输入"Bob",点击提交按钮. 你应该看到 {page04-success.jsp}} 输出"Hello, Bob!"

代码是如何工作的

你大概会去猜测这些代码是如何工作的.

在Action中更改了设置private属性 yourName 的set方法 setYourName() , setParameters() 魔法般的提取出了所有在JSP request 中的对象并把它们放到本地属性Map parameters}}#. ### {{execute() 时, 代替了访问 yourName 变量, 能从 parameters 中得到 "yourName" 字段的值 . So far so good .

回到 page04-success.jsp 页, <ww:property value="yourName" /> 现在不会再工作了, 因为在Acton中已经没有 getYourName() 方法了. 取而代之的,是 <ww:property value="parameters.yourName" /> 调用了{{getParameters()}}方法, 并从中取出 "yourName" 字段的值. 非常漂亮!

我们没有涉及如何处理 radio , checkboxes, 和其它一些html表单组件. That involves dealing with the fact that every entry in the parameters Map is a String[]. We'll cover this in a later lesson.

Understanding interceptors

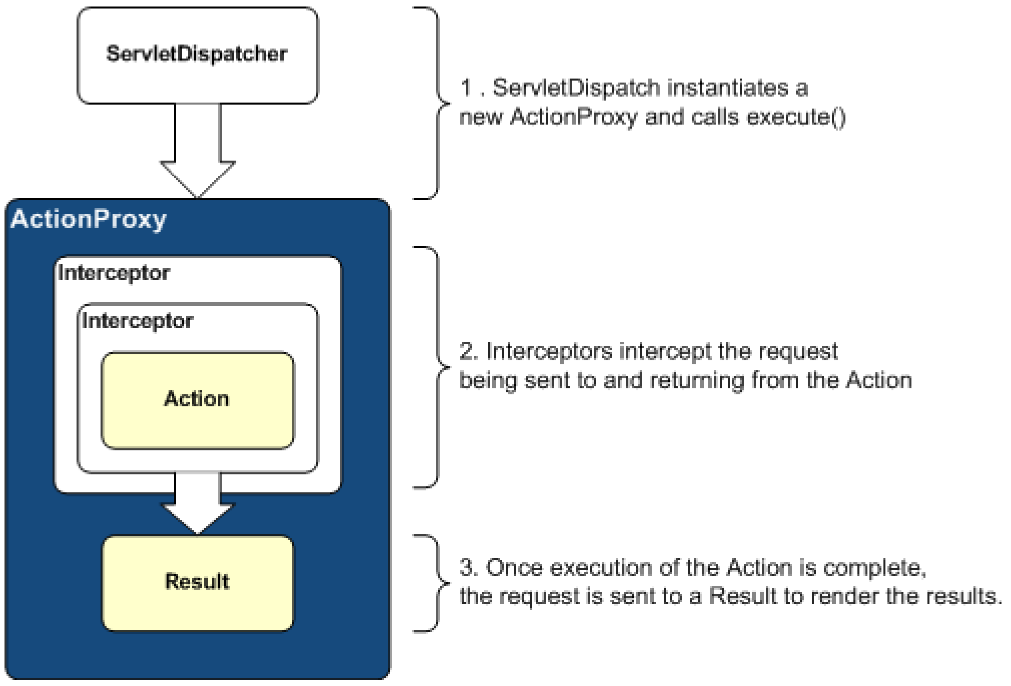
This page last changed on May 17, 2006 by scud.

TODO
Update the info about the new interceptors

Interceptors

Interceptors allow arbitrary code to be included in the call stack for your action before and/or after processing the action, which can vastly simplify your code itself and provide excellent opportunities for code reuse. Many of the features of XWork and WebWork are implemented as interceptors and can be applied via external configuration along with your own Interceptors in whatever order you specify for any set of actions you define.

In other words, when you access a `*.action` URL, WebWork's `ServletDispatcher` proceeds to the invocation of the an action object. Before it is executed, however, the invocation can be intercepted by another object, that is hence called interceptor. To have an interceptor executed before (or after) a given action,


ServletDispatcher
1 . ServletDispatch instantiates a new ActionProxy and calls execute()
ActionProxy
Interceptor
Interceptor
Action
2. Interceptors intercept the request being sent to and returning from the Action
Result
3. Once execution of the Action is complete, the request is sent to a Result to render the results.

Be Careful
Note that some interceptors will interrupt the stack/chain/flow... so the order is very important.

Interceptor configuration:

```
<package name="default" extends="webwork-default">
```

```
    <interceptors>
        <interceptor name="timer" class=".."/>
        <interceptor name="logger" class=".."/>
    </interceptors>

    <action name="login"
       class="org.hibernate.auction.web.actions.users.Login">
         <interceptor-ref name="timer"/>
         <interceptor-ref name="logger"/>
          <result name="input">login.jsp</result>
          <result name="success"
             type="redirect">/secure/dashboard.action</result>
    </action>
 </package>
```

Grouping interceptors as stacks

With most web applications, you'll find yourself wanting to apply the same interceptors over and over. Rather than declare numerous interceptor-refs for each action, you can bundle these interceptors together using an interceptor stack.

```
 <package name="default" extends="webwork-default">
    <interceptors>
         <interceptor name="timer" class=".."/>
         <interceptor name="logger" class=".."/>
         <interceptor-stack name="myStack">
            <interceptor-ref name="timer"/>
            <interceptor-ref name="logger"/>
         </interceptor-stack>
     </interceptors>

 <action name="login"
      class="org.hibernate.auction.web.actions.users.Login">
          <interceptor-ref name="myStack"/>
          <result name="input">login.jsp</result>
          <result name="success"
              type="redirect">/secure/dashboard.action</result>
 </action>
 </package>
```

Looking inside `webwork-default.xml` we can see how it's done:

webwork-default.xml:

```
 <!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
 "http://www.opensymphony.com/xwork/xwork-1.1.dtd">

 <!-- // START SNIPPET: webwork-default -->
 <xwork>
     <package name="webwork-default">
         <result-types>
             <result-type name="chain" class="com.opensymphony.xwork.ActionChainResult"/>
             <result-type name="dispatcher"
 class="com.opensymphony.webwork.dispatcher.ServletDispatcherResult"
                          default="true"/>
             <result-type name="freemarker"
 class="com.opensymphony.webwork.views.freemarker.FreemarkerResult"/>
             <result-type name="httpheader"
 class="com.opensymphony.webwork.dispatcher.HttpHeaderResult"/>
```

```
                <result-type name="jasper"
    class="com.opensymphony.webwork.views.jasperreports.JasperReportsResult"/>
                <result-type name="redirect"
    class="com.opensymphony.webwork.dispatcher.ServletRedirectResult"/>
                <result-type name="redirect-action"
                             class="com.opensymphony.webwork.dispatcher.ServletActionRedirectResult"/>
                <result-type name="stream"
    class="com.opensymphony.webwork.dispatcher.StreamResult"/>
                <result-type name="velocity"
    class="com.opensymphony.webwork.dispatcher.VelocityResult"/>
                <result-type name="xslt" class="com.opensymphony.webwork.views.xslt.XSLTResult"/>
            </result-types>

            <interceptors>
                <interceptor name="alias"
    class="com.opensymphony.xwork.interceptor.AliasInterceptor"/>
                <interceptor name="autowiring"
                             class="com.opensymphony.xwork.spring.interceptor.ActionAutowiringInterceptor"/>
                <interceptor name="chain"
    class="com.opensymphony.xwork.interceptor.ChainingInterceptor"/>
                <interceptor name="component"
    class="com.opensymphony.xwork.interceptor.component.ComponentInterceptor"/>
                <interceptor name="conversionError"
                             class="com.opensymphony.webwork.interceptor.WebWorkConversionErrorInterceptor"/>
                <interceptor name="external-ref"
    class="com.opensymphony.xwork.interceptor.ExternalReferencesInterceptor"/>
                <interceptor name="execAndWait"
    class="com.opensymphony.webwork.interceptor.ExecuteAndWaitInterceptor"/>
                <interceptor name="exception"
    class="com.opensymphony.xwork.interceptor.ExceptionMappingInterceptor"/>
                <interceptor name="fileUpload"
    class="com.opensymphony.webwork.interceptor.FileUploadInterceptor"/>
                <interceptor name="i18n"
    class="com.opensymphony.xwork.interceptor.I18nInterceptor"/>
                <interceptor name="logger"
    class="com.opensymphony.xwork.interceptor.LoggingInterceptor"/>
                <interceptor name="model-driven"
    class="com.opensymphony.xwork.interceptor.ModelDrivenInterceptor"/>
                <interceptor name="params"
    class="com.opensymphony.xwork.interceptor.ParametersInterceptor"/>
                <interceptor name="prepare"
    class="com.opensymphony.xwork.interceptor.PrepareInterceptor"/>
                <interceptor name="static-params"
    class="com.opensymphony.xwork.interceptor.StaticParametersInterceptor"/>
                <interceptor name="servlet-config"
    class="com.opensymphony.webwork.interceptor.ServletConfigInterceptor"/>
                <interceptor name="sessionAutowiring"
                             class="com.opensymphony.webwork.spring.interceptor.SessionContextAutowiringIntercep
                <interceptor name="timer"
    class="com.opensymphony.xwork.interceptor.TimerInterceptor"/>
                <interceptor name="token"
    class="com.opensymphony.webwork.interceptor.TokenInterceptor"/>
                <interceptor name="token-session"
                             class="com.opensymphony.webwork.interceptor.TokenSessionStoreInterceptor"/>
                <interceptor name="validation"
    class="com.opensymphony.xwork.validator.ValidationInterceptor"/>
                <interceptor name="workflow"
    class="com.opensymphony.xwork.interceptor.DefaultWorkflowInterceptor"/>

                <!-- Basic stack -->
                <interceptor-stack name="basicStack">
                    <interceptor-ref name="exception"/>
                    <interceptor-ref name="servlet-config"/>
                    <interceptor-ref name="prepare"/>
                    <interceptor-ref name="static-params"/>
                    <interceptor-ref name="params"/>
                    <interceptor-ref name="conversionError"/>
                </interceptor-stack>

                <!-- Sample validation and workflow stack -->
                <interceptor-stack name="validationWorkflowStack">
                    <interceptor-ref name="basicStack"/>
                    <interceptor-ref name="validation"/>
                    <interceptor-ref name="workflow"/>
                </interceptor-stack>

                <!-- Sample file upload stack -->
                <interceptor-stack name="fileUploadStack">
```

```
                    <interceptor-ref name="fileUpload"/>
                    <interceptor-ref name="basicStack"/>
                </interceptor-stack>

                <!-- Sample WebWork Inversion of Control stack
                     Note: WebWork's IoC is deprecated - please
                     look at alternatives such as Sprint -->
                <interceptor-stack name="componentStack">
                    <interceptor-ref name="component"/>
                    <interceptor-ref name="basicStack"/>
                </interceptor-stack>

                <!-- Sample model-driven stack  -->
                <interceptor-stack name="modelDrivenStack">
                    <interceptor-ref name="model-driven"/>
                    <interceptor-ref name="basicStack"/>
                </interceptor-stack>

                <!-- Sample action chaining stack -->
                <interceptor-stack name="chainStack">
                    <interceptor-ref name="chain"/>
                    <interceptor-ref name="basicStack"/>
                </interceptor-stack>

                <!-- Sample i18n stack -->
                <interceptor-stack name="chainStack">
                    <interceptor-ref name="i18n"/>
                    <interceptor-ref name="basicStack"/>
                </interceptor-stack>

                <!-- Sample execute and wait stack.
                     Note: execAndWait should always be the *last* interceptor. -->
                <interceptor-stack name="executeAndWaitStack">
                    <interceptor-ref name="basicStack"/>
                    <interceptor-ref name="execAndWait"/>
                </interceptor-stack>

                <!-- A complete stack with all the common interceptors in place.
                     Generally, this stack should be the one you use, though it
                     may process additional stuff you don't need, which could
                     lead to some performance problems. Also, the ordering can be
                     switched around (ex: if you wish to have your components
                     before prepare() is called, you'd need to move the component
                     interceptor up -->
                <interceptor-stack name="defaultStack">
                    <interceptor-ref name="exception"/>
                    <interceptor-ref name="alias"/>
                    <interceptor-ref name="prepare"/>
                    <interceptor-ref name="servlet-config"/>
                    <interceptor-ref name="i18n"/>
                    <interceptor-ref name="chain"/>
                    <interceptor-ref name="model-driven"/>
                    <interceptor-ref name="fileUpload"/>
                    <interceptor-ref name="static-params"/>
                    <interceptor-ref name="params"/>
                    <interceptor-ref name="conversionError"/>
                    <interceptor-ref name="validation"/>
                    <interceptor-ref name="workflow"/>
                </interceptor-stack>

                <!-- The completeStack is here for backwards compatibility for
                     applications that still refer to the defaultStack by the
                     old name -->
                <interceptor-stack name="completeStack">
                    <interceptor-ref name="defaultStack"/>
                </interceptor-stack>
            </interceptors>

            <default-interceptor-ref name="defaultStack"/>
        </package>
    </xwork>
    <!-- // END SNIPPET: webwork-default -->
```

Since we included `webwork-default.xml` in our `xwork.xml`, all the interceptors and stacks above are available for us to use in our actions. Here's what these interceptors do:

- `timer`: clocks how long the action (including nested interceptors and view) takes to execute;
- `logger`: logs the action being executed;
- `chain`: makes the previous action's properties available to the current action. Used to make action chaining (reference: Result Types);
- `static-params`: sets the parameters defined in `xwork.xml` onto the action. These are the `<param />` tags that are direct children of the `<action />` tag;
- `params`: sets the request (POST and GET) parameters onto the action class. We have seen an example of this in TODO;
- `model-driven`: if the action implements `ModelDriven`, pushes the `getModel()` result onto the Value Stack;
- `component`: enables and makes registered components available to the actions. (reference: IoC & Components);
- `token`: checks for valid token presence in action, prevents duplicate form submission;
- `token-session`: same as above, but storing the submitted data in session when handed an invalid token;
- `validation`: performs validation using the validators defined in `{Action}-validation.xml` (reference: Validation).;
- `workflow`: calls the validate method in your action class. If action errors created then it returns the `INPUT` view. Good to use together with the validation interceptor (reference: Validation);
- `servlet-config`: give access to `HttpServletRequest` and `HttpServletResponse` (think twice before using this since this ties you to the Servlet API);
- `prepare`: allows you to programmatic access to your Action class before the parameters are set on it.;
- `conversionError`: Adds field errors if any type-conversion errors occurred.
- `execAndWait`: Spawns a separate thread to execute the action
- `fileUpload`: Sets uploaded files as action files (File objects)

In addition to the prepackaged interceptors, webwork-default.xml includes prepackaged combinations of these interceptors in named interceptor stacks.

Building your own Interceptor

If none of the above interceptors suit your particular need, you will have to implement your own interceptor. Fortunately, this is an easy task to accomplish. Suppose we need an interceptor that places a greeting in the Session according to the time of the day (morning, afternoon or evening). Here's how we could implement it:

1. Create an interceptor class, which is a class that implements the `com.opensymphony.xwork.interceptor.Interceptor` interface (bundled in `xwork-1.1.jar`);
2. Declare the class in your XML configuration file (`xwork.xml`) using the element `<interceptor />` nested within `<interceptors />`;
3. Create stacks of interceptors, using the `<interceptor-stack />` element (optional);
4. Determine which interceptors are used by which action, using `<interceptor-ref />` or `<default-interceptor-ref />`. The former defines the interceptors to be used in a specific action, while the latter determines the default interceptor stack to be used by all actions that do not specify their own `<interceptor-ref />`.

GreetingInterceptor.java:

```
package lesson05;

import java.util.Calendar;
import com.opensymphony.xwork.interceptor.Interceptor;
import com.opensymphony.xwork.ActionInvocation;

public class GreetingInterceptor implements Interceptor {
        public void init() { }
        public void destroy() { }
        public String intercept(ActionInvocation invocation) throws Exception {
                Calendar calendar = Calendar.getInstance();
                int hour = calendar.get(Calendar.HOUR_OF_DAY);
                String greeting = (hour < 6) ? "Good evening" :
                        ((hour < 12) ? "Good morning":
                        ((hour < 18) ? "Good afternoon": "Good evening"));

                invocation.getInvocationContext().getSession().put("greeting", greeting);

                String result = invocation.invoke();

                return result;
        }
}
```

xwork.xml:

```
<!DOCTYPE xwork PUBLIC "-//OpenSymphony Group//XWork 1.0//EN"
"http://www.opensymphony.com/xwork/xwork-1.0.dtd">

<xwork>
        <!-- Include webwork defaults (from WebWork JAR). -->
        <include file="webwork-default.xml" />

        <!-- Configuration for the default package. -->
        <package name="default" extends="webwork-default">
                <interceptors>
                        <interceptor name="greeting"
class="section02.lesson05.GreetingInterceptor" />
                </interceptors>

                <!-- Action: Lesson 5: GreetingInterceptor. -->
                <action name="greetingAction" class="lesson05.GreetingAction">
                        <result name="success" type="velocity">ex01-result.vm</result>
                        <interceptor-ref name="greeting" />
                </action>
        </package>
</xwork>
```

GreetingAction.java:

```
package lesson05;

import com.opensymphony.xwork.ActionSupport;

public class GreetingAction extends ActionSupport {
        public String execute() throws Exception {
                return SUCCESS;
        }
}
```

ex01-result.vm:

```
<html>
<head>
<title>WebWork Tutorial - Lesson 5 - Example 1</title>
</head>
<body>

#set ($ses = $req.getSession())
<p><b>${ses.getAttribute('greeting')}!</b></p>

</body>
</html>
```

Let's take a look at our interceptor class first. As explained before, the interceptor must implement `com.opensymphony.xwork.interceptor.Interceptor`'s methods: `init()`, called during interceptor initialization, `destroy()`, called during destruction, and most importantly, `intercept(ActionInvocation invocation)`, which is where we place the code that does the work.

Notice that our interceptor returns the result from `invocation.invoke()` which is the method responsible for executing the next interceptor in the stack or, if this is the last one, the action. This means that the interceptor has the power of short-circuiting the action invocation and return a result string without executing the action at all! Use this with caution, though.

One other thing that interceptors can do is execute code after the action has executed. To do that, just place code after the invocation.invoke() call. WebWork provides an abstract class that already implements this kind of behaviour: `com.opensymphony.xwork.interceptor.AroundInterceptor`. Just extend it and implement the methods `before(ActionInvocation invocation)` and `after(ActionInvocation dispatcher, String result)`.

The `xwork.xml` configuration, the action class and the result page are pretty straightforward and require no further explanation.

Try the example\!

Understanding results

This page last changed on May 13, 2006 by dogg.

一个Result是你的action中部分代码执行完成之后的返回值，比如 success 或 error. WebWork几乎包含你所需要的大部分result,例如: 像"servlet dispatcher," 已经使用过的JSPs, 和Velocity 以及诸如其它FreeMarker和Jasper Reports (输出 PDF, XML,和HTML).

```
<action name="form03" class="lessons.Form03Action">
      <result name="success" type="dispatcher">page03-success.jsp</result>
      <result name="error" type="dispatcher">page03-error.jsp</result>
</action>
```

正像你看到的那样, result配置是由action 映射和result类型两部分的联合装配起来的.

配置result类型

在WebWork 中的每一个包都能关联一个或多个result类型.

```
<xwork>
   <include name="webwork-default.xml"/>
   <package name="default" extends="webwork-default">

      <result-types>
        <result-type name="dispatcher" class="..." default="true"/>
        <result-type name="redirect" class="..."/>
      </result-types>

      <default-interceptor-ref name="defaultStack"/>

      <action name="login"
           class="org.hibernate.auction.web.actions.users.Login">
        <result name="input">login.jsp</result>
        <result name="success"
           type="redirect">/secure/dashboard.action</result>
     </action>
   </package>
</xwork>
```

使用全局result映射减少重复的(duplication)配置

在 xwork.xml 中减少配置数量的方法是使用全局result映射. Web应用中常常有许多Action拥有共同的Results定义. 共同的Results 包含重定向到登陆Actions和拒绝访问页. WebWork可以让你集中定义那些公共的页面，而无须在每一个Action映射中定义result.

```
<package name="default" extends="webwork-default">
   <global-results>
      <result name="login"
         type="redirect">/login!default.action</result>
      <result name="unauthorized">/unauthorized.jsp</result>
   </global-results>
   <!-- other package declarations -->
</package>
```

⚠ **要小心**

因为全局reseults搜索顺序在局部results之后,当你为特定的Action定义了result后就能覆盖任意的全局result映射.记得Results的接入位置可以使用绝对或相对路径.因为你可能不知道调用上下文的情况下,它们被引用,所以

在全局Results定义中最好使用绝对路径.

Understanding validation

This page last changed on Apr 12, 2006 by dogg.

缺少关于此标题的文章,你可以查看一篇非常好的文章了解信息 Webwork Validation在java.net上.

Type Conversion

This page last changed on May 19, 2006 by scud.

在所有的基于Web的Java开发框架中, WebWork拥有最优秀的类型转换能力. 通常情况下, 要利用这种能力, 只需要把HTML输入项(表单元素和其他GET/POST的参数)命名为合法的OGNL表达式.

简单的例子

当你需要将一个字符串转换成为一个更为复杂的对象时, 类型转换能发挥强大的作用. 由于Web是类型不可知的(type-agnostic)(HTTP中任何类型都是字符串), 因此WebWork的类型转换功能非常有用. 例如, 如果你提示用户使用字符串格式("3, 22")输入一个坐标, 你需要让WebWork完成String到Point和Point到String的转换.

使用"point"的例子, 如果活动(或者其他需要设置属性的组合对象)有一个对应的ClassName-conversion.properties文件, WebWork会使用配置的类型转换器进行该类和字符串的转换. 因此将"3, 22"转换成new Point(3, 22)只需在 `ClassName-conversion.properties` 中加入下列内容(注意, PointConverter需要实现接口ognl.TypeConverter):

```
point = com.acme.PointConverter
```

你编写的类型转换器必须检查被用于转换哪种类型. 由于都是与String进行相互转换, 需要将转换方法分为两部分: 一个将String转换为Point, 另一个将Point转换为String.

这些完成后, 你可以引用point对象(在JSP中使用<ww:property value="point"/>或在FreeMarker中使用${point})并且它将输出为"3, 22". 同样的, 如果你将它提交到活动, 它将再次转换成Point.

有时你可能希望将一个类型转换器应用到全局范围. 这可以通过编辑 `xwork-conversion.properties` (位于根类路径, 通常是WEB-INF/classes目录)并以下列形式添加一个属性定义: 左边是你希望转换的类名称, 右边是转换器的类名称. 例如, 为所有的Point对象提供一个类型转换器意味着增加下列内容:

```
com.acme.Point = com.acme.PointConverter
```

(摘自snippet:id=javadoc|javadoc=true|url=com.opensymphony.xwork.util.XWorkConverter)

⚠ 类型转换不能作为i18n的替代品. 不推荐使用这一特性输出相应的日期格式. 相反, 你应该使用WebWork的i18n功能(并且研究一下JDK中MessageFormat的JavaDoc)来了解如何使用相应正确的日期格式.
(摘自snippet:id=i18n-note|javadoc=true|url=com.opensymphony.xwork.util.XWorkConverter)

WebWork附带了一个帮助基类(a helper base class), 利用它可以很方便的完成类型与字符串之间的相互转换. 这个类就是 `com.opensymphony.webwork.util.WebWorkTypeConverter`. 该类可以让你很方便的编写处理对象和字符串相互转换的类型转换器. 下面是从该类的JavaDoc中摘录的内容:

本类是WebWork使用的类型装换器的一个基类. 该类提供了两个抽象方法用于对象与字符串的相互转换 - 这是WebWork类型转换系统的关键功能.

类型转换器也可以不使用该类. 它基本上是一个帮助类, 尽管它是推荐使用的, 它为子类提供了所有基于Web的类型转换所需的一般接口约束.

它还包含一个hook(后备方法), 名为performFallbackConversion, 该方法可用于在convertValue方法执行失败时执行一些后备的转换. 缺省情况下, 它使用父类(Ognl的DefaultTypeConverter)的convertValue方法完成转换.

(摘自snippet:id=javadoc|javadoc=true|url=com.opensymphony.webwork.util.WebWorkTypeConverter)

内建的(Built in)类型转换支持

WebWork可以替你自动完成大多数常用的类型转换. 已支持的与字符串之间转换类型包括:

- String
- boolean / Boolean
- char / Character
- int / Integer, float / Float, long / Long, double / Double
- dates - 使用当前request指定的Locale信息对应的SHORT格式
- arrays - 假定每一个字符串都能够转换成对应的数组元素
- collections - 如果不能确定对象类型, 将假定集合元素类型为String, 并创建一个新的ArrayList

注意, 对于数组的类型转换将按造数组元素的类型来单独转换每一个元素. 而在其他类型转换中, 如果转换无法实现, 将使用标准的类型转换错误报告.

(摘自snippet:id=javadoc|javadoc=true|url=com.opensymphony.xwork.util.XWorkBasicConverter)

参数名称的关系

利用WebWork的类型转换最好的方式是使用已完成的(completes)对象(理想情况下应当直接使用业务对象(domain objects)), 而不是使用基本类型或字符串类型的表单参数值作为中间值, 然后在Action的execute()方法中把这些中间值转换成完整的对象(rather than submitting form values on to intermediate primitives and strings in your action and then converting those values to full objects in the execute() method). 下面是一些提示:

- 使用组合的(complex)OGNL表达式 - WebWork能自动创建你所需的实际对象.
- 使用JavaBeans! WebWork只能创建遵守JavaBean规范的对象, 这需要你的对象提供一个无参构造函数, 并包含适当的getter和setter方法.
- 记住 person.name 将调用 **getPerson().setName()**, 但如果你希望WebWork创建Person对象, 那么**必须包含一个 setPerson() 方法**.
- 对于list和map对象, 使用索引符号, 如 people[0].name or friends['patrick'].name. 通常这些HTML表单元素是在一个循环中绘制出来的, 因此你可以在JSP Tags中使用iterator标签的状态属性(status attribute)或在FreeMarker Tags中使用${foo_index} 来指定这一属性(指index value, 译注).
- 对于多选的列表, 显然不能为每个单独的选项使用对应的属性符号来命名(由于). 替代的方法是, 使用简单的名称 people.name 来命名你的表单元素, WebWork知道需要为每一个选中的选项创建一个新的Person对象并设定它的名字.

高级类型转换

WebWork还有一些非常优秀的(同样易与使用)的类型转换特性. 对空值(Null)属性的处理可以在发现空值引用时自动创建对象. 对Collection和Map的支持提供了针对Java集合的智能空值处理和类型转换. 类型转换错误处理提供了一种简单的方法可以把输入校验问题和输入类型装换问题区别开.

空值属性处理

通过把action context中的键值 `CREATE_NULL_OBJECTS` 设置为true支持空值处理(该键只在com.opensymphony.xwork.interceptor.ParametersInterceptoris执行过程中进行设置), 这样, 出现NullPointerException异常的OGNL表达式将被自动临时中断, 然后系统将通过创建所需对象的方法来自动尝试解决null引用.(OGNL中包含了NullHandler接口, 允许对空值引用进行转换, WebWork实现了这一接口并替换了OGNL的缺省实现, 译注)

处理控制引用时将遵循下列规则:

- 如果属性声明为Collection或List，将返回一个ArrayList并赋值给空引用.
- 如果属性声明为Map, 将返回一个HashMap并赋值给空引用.
- 如果空值属性是一个带有无参构造函数的简单Bean, 将使用ObjectFactory.buildBean(java.lang.Class, java.util.Map)方法创建一个实例.

(摘自snippet:id=javadoc|javadoc=true|url=com.opensymphony.xwork.util.InstantiatingNullHandler)

例如, 如果表单中包含一个文本字段名为 person.name 而表达式 person 运算结果为null, 那么该类将被调用. 由于表达式 person 类型为Person, 因此将创建一个新的Person实例并赋值给空值引用. 最后, name值被赋给该实例的name属性. 全部过程是系统自动创建一个Person实例, 并调用setPerson()方法将它赋给空值引用, 最后调用getPerson().setName(), 而这通常是想要的结果.

(摘自snippet:id=example|javadoc=true|url=com.opensymphony.xwork.util.InstantiatingNullHandler)

Collection和Map支持

WebWork支持多种方法来判断集合中的对象类型. 这是通过一个 ObjectTypeDeterminer 完成的. WebWork提供了缺省实现. 下面的JavaDocs解释了对Map和Collection的支持是如何在DefaultObjectTypeDeterminer中完成的:

ObjectTypeDeterminer检查 Class-conversion.properties 文件中包含的用于表示Map和Collection中包含的对象类型的相关内容 . 对于Collection, 如List, 使用格式 Element_xxx 来指定其中的元素类型, 这里xxx是action或其他对象中的集合属性名称. 对于Map, 需要按照格式 Key_xxx 和 Element_xxx 分别指定key和value的类型.

从WebWork 2.1.x开始, 仍然支持Collection_xxx这样的书写格式, 尽管这种格式已经被声明废弃而且最终将被去除.

(摘自snippet:id=javadoc|javadoc=true|url=com.opensymphony.xwork.util.DefaultObjectTypeDeterminer)

除此之外, 也可以实现接口ObjectTypeDeterminer来创建自己的定制ObjectTypeDeterminer. WebWork也包含一个可选的使用Java5范型(generics)技术实现的ObjectTypeDeterminer. 更多信息参见J2SE 5 Support.

使用集合的属性索引集合元素(Indexing a collection by a property of that collection)

也可以向WebWork传递元素的某个给定属性的值来获取集合中的唯一元素(element). 缺省情况下, 这个属性由Class-conversion.properties中定义的KeyProperty_xxx=yyy决定, 这里的xxx是返回集合的JavaBean类型名称, yyy是我们用于索引集合中元素的属性名称. (the property of the element of the collection is determined in Class-conversion.properties using KeyProperty_xxx=yyy where xxx is the property of the bean 'Class' that returns the collection and yyy is the property of the collection element that we want to index on.) 下面的两个类是一个示例:

```
/**
 * @return a Collection of Foo objects
 */
public Collection getFooCollection()
{
    return foo;
}
```

```
/**
 * @return a unique identifier
 */
public Long getId()
{
    return id;
}
```

然后将 KeyProperty_fooCollection=id 放在MyAction-conversion.properties文件中. 这样就可以使用 fooCollection(someIdValue) 从集合fooCollection中获取id等于 someIdValue 的Foo对象. 例如, fooCollection(22) 将得到id值为22的Foo对象.

这一点十分有用, 因为这直接将一个集合中的元素与它的唯一标志符联系起来, 而不需要强制使用索引, 从而允许修改一个Bean的集合中的元素而不需要编写额外的代码. 例如, 值为 Phil 的参数 fooCollection(22).name 将集合fooClooection中id属性值为22的Foo对象的name属性设置为"Phil".

Webwork可以使用类型转换自动将参数的类型转换成key属性的类型.

与Map和List元素的属性不同, 如果fooCollection(22)不存在, WebWork不会创建新的对象. 想要做到这一点, 可以使用符号 fooCollection.makeNew[index], 在这里index是一个整数(0, 1等等). 因此, 参数 fooCollection.makeNew[0]=Phil 以及 fooCollection.makeNew[1]=John 将在fooCollection中添加两个新的Foo对象, 一个name属性值为Phil, 另一个为Bar. 注意, 不管用哪种方法, 在使用Set类型时, 必须定义对象的equals方法和hashCode方法来并保证他们不仅仅包含id属性. 这将导致id属性为null的元素可以从Set中删除.

索引的List和Map的高级示例

下面是一个用于List中的模型bean.
该类的KeyProperty是id属性.

```
public class MyBean implements Serializable {

    private Long id;
    private String name;

    public Long getId() {
        return id;
    }

    public void setId(Long id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }


    public String toString() {
        return "MyBean{" +
                "id=" + id +
                ", name='" + name + '\'' +
                '}';
    }
}
```

下面的action有一个beanList属性, 它被初始化为一个空的ArrayList对象.

```
ublic class MyBeanAction implements Action {

    private List beanList = new ArrayList();
    private Map beanMap = new HashMap();

    public List getBeanList() {
        return beanList;
    }
```

```
        public void setBeanList(List beanList) {
            this.beanList = beanList;
        }

        public Map getBeanMap() {
            return beanMap;
        }

        public void setBeanMap(Map beanMap) {
            this.beanMap = beanMap;
        }

        public String execute() throws Exception {
            return SUCCESS;
        }
    }
```

定义在conversion.properties中的内容告诉TypeConverter使用MyBean的实例作为List的元素.

```
KeyProperty_beanList=id
Element_beanList=MyBean
CreateIfNull_beanList=true
```

当通过表单提交到Action时, (id)的值被当作beanList中MyBean实例的KeyProperty.
注意()符号! 不要使用[], 那仅能用在Map里!
属性的name的值将根据指定的id设置到对应的MyBean实例中.
无效的id值将不会再加入null值.
这就避免了OutOfMemory异常的产生!

```
<ww:iterator value="beanList" id="bean">
  <ww:textfield name="beanList(%{bean.id}).name" />
</ww:iterator>
```

类型转换错误处理

当类型转换期间发生错误时, 有时希望报告这些错误, 而有时不希望报告. 例如, 报告输入的"abc"不能转换成数字可能很重要. 另一方面, 报告一个空字符串("")不能装换成数字可能不重要 - 除非是在一个Web环境下, 难以区分用户没有输入值还是输入了一个空白值.

缺省情况下, 所有的装换错误使用通用的i18n信息 xwork.default.invalid.fieldvalue , 你可以在你的全局i18n资源包中替换他(缺省文本是"Invalid field value for field xxx", 这里xxx是字段名称).

无论如何, 有时你会希望能够在每个字段上替换这一信息. 你可以在action相关的资源文件(Action.properties)中添加一个i18n信息: invalid.fieldvalue.xxx, 这里xxx是字段名称.

需要知道的是, 这些错误不会直接报告出来. 他们被添加到ActionContext.conversionErrors中. 有几种方法可以访问该map从而可以报告这些错误.

(摘自snippet:id=error-reporting|javadoc=true|url=com.opensymphony.xwork.util.XWorkConverter)

错误报告可以两种方式处理:

1. 全局方式, 使用 Conversion Error Interceptor 报告错误
2. 以每一个字段为基点, 使用 conversion validator 报告错误

缺省情况下, conversion interceptor包含在 webwork\-default.xml 的缺省截取器栈中, 如果不希望使用全局错误报告

方式，需要修改截取器栈并添加其他的校验规则(指第二种方式，在字段上使用conversion validator，译注).

Validation

This page last changed on May 20, 2006 by scud.

WebWork依赖XWork的校验框架,在你的action被执行之前,来启用对你的action的输入数据的校验规则的运用.本章节仅提供了最基本的内容来让你上手,并集中精力在WebWork为了支持客户端校验对XWork的校验器的扩展上.

对于基于客户端的校验(JavaScript 和/或 AJAX) 还有一个选项,请浏览 客户端校验 了解更多信息.

示例

1. 基本校验
2. 客户端校验
3. AJAX校验
4. 使用字段校验器
5. 使用非字段校验器
6. 使用Visitor字段校验器

WebWork 里可用的校验器

注意

当使用一个字段校验器的时候,字段校验器语法一般回避使用普通的校验器语法更好,因为它更宜于按照字段对字段校验器进行分组.这是非常方便的,特别是一个字段需要有多个字段校验器的时候,而这种情况会经常发生.例如: validatortypes

1. 必填校验器
2. 必填字符串校验器
3. 整数校验器
4. 日期校验器
5. 表达式校验器
6. 字段表达式校验器
7. 邮件校验器
8. 网址校验器
9. visitor校验器
10. 转换校验器
11. 字符串长度校验器
12. 正则表达式校验器

注册校验器

校验规则是通过校验器来处理的,它们必须注册到 ValidatorFactory (使用 registerValidator 方法).最简单的方法就是添加一个文件名为 validators.xml 的文件在你的classpath (/WEB-INF/classes) 的根目录下,来声明所有你想使用的校验器.

这个列表声明了所有WebWork带来的校验器.

```
<validators>
    <validator name="required"
class="com.opensymphony.xwork.validator.validators.RequiredFieldValidator"/>
    <validator name="requiredstring"
class="com.opensymphony.xwork.validator.validators.RequiredStringValidator"/>
    <validator name="int"
class="com.opensymphony.xwork.validator.validators.IntRangeFieldValidator"/>
    <validator name="double"
class="com.opensymphony.xwork.validator.validators.DoubleRangeFieldValidator"/>
    <validator name="date"
class="com.opensymphony.xwork.validator.validators.DateRangeFieldValidator"/>
    <validator name="expression"
class="com.opensymphony.xwork.validator.validators.ExpressionValidator"/>
    <validator name="fieldexpression"
class="com.opensymphony.xwork.validator.validators.FieldExpressionValidator"/>
    <validator name="email"
class="com.opensymphony.xwork.validator.validators.EmailValidator"/>
    <validator name="url" class="com.opensymphony.xwork.validator.validators.URLValidator"/>
    <validator name="visitor"
class="com.opensymphony.xwork.validator.validators.VisitorFieldValidator"/>
    <validator name="conversion"
class="com.opensymphony.xwork.validator.validators.ConversionErrorFieldValidator"/>
    <validator name="stringlength"
class="com.opensymphony.xwork.validator.validators.StringLengthFieldValidator"/>
    <validator name="regex"
class="com.opensymphony.xwork.validator.validators.RegexFieldValidator"/>
</validators>
```

注意

validators.xml如果已经定义了,那么它应该在classpath中可以找到.然而如果不需要自定义的校验器,那么这不是必须的.WebWork会自动从发布包里的xwork jar文件中取得一个事先定义好的校验器集合(com/opensymphony/xwork/validator/validators/default.xml). 浏览ValidatorFactory的static块来了解详细信息.

警告

如果自定义的校验器被定义了而且创建了一个validators.xml文件并放在classpath中,记得复制所有其他你需要的预定义的校验器到validators.xml里,如果你不需要注册则不需要.一旦validators.xml在classpath里被检测到,缺省的 (com/opensymphony/xwork/validator/validators/default.xml)就不会被装载了.只有没发现自定义validators.xml的时候才会装载.要小心.

打开校验

缺省的 validationWorkflowStack 已经包含了这个.
为了开启一个action的校验所有需要做的就是在action的拦截器ref中包含ValidationInterceptor (浏览 xwork.xml),就像这样:

```
<interceptor name="validator" class="com.opensymphony.xwork.validator.ValidationInterceptor"/>
```

注意: 缺省的 validationWorkflowStack 已经包含了这个拦截器.

校验器会话范围

字段校验器,就像名字暗示的那样,作用在通过action可以访问的单个字段上.一个校验器(译注:怀疑为 "一个非字段校验器"),与字段校验器相反,是一个更普通的,能在全部的action上下文中进行校验,可以在校验规则中包含多于一个字段(甚至根本不是字段).大多数的校验可以基于每个字段的基础上定义.只要可能字段校验器相对非字段校验器都是首选的,字段校验器的信息是绑定到相关的字段上的,并会在对应的视图(页面)中显示在对应输入元素的周围.

非字段校验器仅会添加action级别的信息.非字段校验器经常是domain相关的因为经常是自定义的实现.XWork/WebWork提供的最重要的标准非字段校验器是 ExpressionValidator .

注意事项

非字段校验器优先于字段校验器,而不管在 *-validation.xml 里是如何定义的.如果一个非字段校验器是短路的,它会引发它的非字段校验器不会被执行.查看校验框架的文档了解更多信息.

定义校验规则

校验规则可以被指定:

1. 每个Action类: 在一个 ActionName-validation.xml 文件中
2. 每个Action的别名: 在一个 ActionName-alias-validation.xml 文件中
3. Action类的继承层次关系和实现的接口: WebWork 搜索action的层次关系树来来查找action的父类和实现的接口的缺省校验

这是一个 SimpleAction-validation.xml 的例子:

```
<!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
       "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
<validators>
  <field name="bar">
      <field-validator type="required">
          <message>You must enter a value for bar.</message>
      </field-validator>
      <field-validator type="int">
          <param name="min">6</param>
          <param name="max">10</param>
          <message>bar must be between ${min} and ${max}, current value is ${bar}.</message>
      </field-validator>
  </field>
  <field name="bar2">
      <field-validator type="regex">
          <param name="regex">[0-9],[0-9]</param>
          <message>The value of bar2 must be in the format "x, y", where x and y are between 0
and 9</message>
     </field-validator>
  </field>
  <field name="date">
      <field-validator type="date">
          <param name="min">12/22/2002</param>
          <param name="max">12/25/2002</param>
          <message>The date must be between 12-22-2002 and 12-25-2002.</message>
      </field-validator>
  </field>
  <field name="foo">
      <field-validator type="int">
          <param name="min">0</param>
          <param name="max">100</param>
```

```
            <message key="foo.range">Could not find foo.range!</message>
        </field-validator>
    </field>
    <validator type="expression">
        <param name="expression"> foo &gt; bar</param>
        <message>Foo must be greater than Bar. Foo = ${foo}, Bar = ${bar}.</message>
    </validator>
  </validators>
```

这里我们可以看到SimpleAction类的校验器配置.校验器(和字段校验器)必须有一个type属性,指向在前面说的ValidatorFactory 中注册的校验器的名字.validators元素还必须有<param>元素,带有name和value属性来设置校验器实例的任何需要设置的参数.浏览下面的文字来讨论message元素.

每个Validator 或者 Field-Validator 元素必须在validator元素体内定义一个message元素.这个message元素有一个属性,key,它不是必填的.message标签的body会被当作缺省的信息,如果校验失败它会被添加到Action中.Key用来在Action的资源包里作为寻找一个信息的主键,如果Action实现了LocaleAware接口(ActionSupport已经实现了这个接口),那么会使用getText()来获取本地化信息.这个功能基于用户发起请求的Locale(或者任何你设置到实现了LocaleAware的Action中的Locale)来提供本地化信息.在不管是使用Key值从资源包中获取信息还是使用缺省的信息,当前的校验器被推送ValueStack中,然后消息被解析来处理其中的 \${...} 部分, \${ 和 } 中间的部分会被求值然后替换相应的部分.这允许你使用校验器,Action或者两者兼有的值来参数化你的信息.

如果校验器失败了,校验器会被推送到ValueStack中,信息也会,不管是缺省的还是本地化的信息(如果key属性被指定,而且存在对应的信息),消息被解析来处理其中的 \${...} 部分, \${ 和 } 中间的部分会被求值然后替换相应的部分.这允许你使用校验器,Action或者两者兼有的值来参数化你的信息.

注意
因为校验规则是在一个XML文件里,你必须要转义特殊字符.例如,注意上面的expression校验器规则,我们使用 ">" 来代替">".参考一个XML相关的文档来了解必须转义的字符的全部列表.最常使用的必须转义的字符是: & (使用 &amp;), > (使用 &gt;) 和 < (使用 &lt;).

这是一个参数化信息的例子:

这会把 IntRangeFieldValidator 的 min和max参数以及Action中的bar的值设置到信息里.

```
  bar must be between ${min} and ${max}, current value is ${bar}.
```

为了给一个Action定义校验规则,在Action相同的包下面创建一个 ActionName-validation.xml 的文件.你也可能需要创建一个别名特定的校验规则,它会添加在ActionName-validtion.xml里面定义的缺省校验规则,只要在相同的目录下创建一个ActionName-aliasName-validation.xml名称的文件就可以.(译注:此段翻译的可能有点问题.使用别名校验时也会使用缺省的校验规则校验) 在两种情况下,ActionName都是Action类的名字,aliasName是在Action的配置文件里xwork.xml里面定义的Action的别名.

框架也会搜索Action的层次关系树来查找Action的父类或者实现的接口的校验规则.当联合使用 ModelDriven Action和 VisitorFieldValidator 时这特别强大.这是一个校验规则如何发现的例子.给出下面的类结构:

- 接口 Animal;
- 接口 Quadraped 扩展了 Animal;
- 类 AnimalImpl 实现了 Animal;
- 类 QuadrapedImpl 扩展了 AnimalImpl 实现了 Quadraped;
- 类 Dog 扩展了 QuadrapedImpl;

如果Dog要被校验,框架方法会查找下面的配置文件:

- Animal
- Animal-aliasname
- AnimalImpl
- AnimalImpl-aliasname
- Quadraped
- Quadraped-aliasname
- QuadrapedImpl
- QuadrapedImpl-aliasname
- Dog
- Dog-aliasname

这个过程类似XWork的本地化框架查找信息时的过程,也有一些小小的不同.最重要的区别时校验规则是自上而下的.

注意: 按照上面的类层次关系的定义,子节点的 *-validation.xml 会覆盖父接口 *-validation.xml .

校验器风味(Flavour)

XWork发布包提供的校验器(以及你可能自己编写的任何校验器)分为两种风味:

1. 普通校验器 /非字段校验器
2. 字段校验器

简单的校验器(例如ExpressionValidator)执行校验检查不是绑定到单一的指定的字段上的.当你在你的 -validation.xml 文件中声明一个简单的校验器时,你不需要把一个fieldname属性和它关联(你应该避免使用下面描述的语法来使用简单校验器).

字段校验器(例如EmailValidator)设置用来在一个单一字段上进行校验检查.它们要求你在你的 -validation.xml 文件中执行一个fieldname属性.有两种不同的XML语法(但是等效)你可以用来声明字段校验器(浏览下面的 " Vs " 的语法).

两种风味的校验器有两个地方有重要的区别需要记住:

1. 当选择xml语法用来声明一个校验器 (或者)
2. 当使用 short-circuit 特性

注意: 注意你不需要在你的 -validation.xml 文件里声明你使用了那种 "风味",你仅仅需要声明要使用的校验器的名字 ,WebWork会知道它是一个普通的校验器还是一个字段校验器,这通过查看校验器编写者选择实现的校验类来实现.

非字段校验器 Vs 字段校验器

有两种方法你可以在你的 -validation.xml 文件里定义校验器:

1. <validator>
2. <field-validator>

无论使用那种语法都要记住后面的内容:

非字段校验器 <validator>元素允许你声明两种校验器的任何一种(普通校验器或者字段相关的字段校验器).

```
<!-- Declaring a plain Validator using the
 <validator> syntax: -->

   <validator type="expression>
         <param name="expression">foo &gt; bar</param>
         <message>foo must be great than bar.</message>
   </validator>
```

```
<!-- Declaring a field validator using the
 <validator> syntax; -->

   <validator type="required">
        <param name="fieldName">bar</param>
        <message>You must enter a value for bar.</message>
   &lt/validator>
```

字段校验器 <field-validator> 元素基本和<validator>元素相同,除了它们从封闭的<field>元素集成了fieldName属性. 使用一个 <field-validator> 元素定义的字段校验器的fieldName属性会自动被它们的父节点<field>元素的fieldName属性填充.这个结构的理由事方便地为一个特定的字段在一个元素下对校验器进行分组,否则fieldName属性会必须被重复,一次又一次,在每个单独的<validator>上.

提示: 在一个<field>标签内定义字段校验器比使用带有一个fieldName参数的<validator>标签好的多,而且xml代码本身也清晰的多(字段分组更清晰了)

注意: 注意你应该在一个块内仅使用一个字段校验器(不是普通校验器).一个<field>内的普通校验器是不允许的,解析xml的时候就会发生错误,因为它在定义的dtd (xwork-validator-1.0.2.dtd) 内是不允许的.

使用<field-validator>语法来声明一个字段校验器:

```
<field name="email_address">
  <field-validator type="required">
      <message>You cannot leave the email address field empty.</message>
  </field-validator>
  <field-validator type="email">
      <message>The email address you entered is not valid.</message>
  </field-validator>
</field>
```

选择权在你手里.仅使用没有任何元素的校验器并且为它们的每一个设置fieldName属性是完全合法的.(It's perfectly legal to only use elements without the elements and set the fieldName attribute for each of them.) 下面的两种方式是等效的:

```
<field name="email_address">
  <field-validator type="required">
      <message>You cannot leave the email address field empty.</message>
  </field-validator>
  <field-validator type="email">
      <message>The email address you entered is not valid.</message>
  </field-validator>
</field>
```

```
<validator type="required">
  <param name="fieldName">email_address</param>
  <message>You cannot leave the email address field empty.</message>
</validator>
<validator type="email">
  <param name="fieldName">email_address</param>
```

```
    <message>The email address you entered is not valid.</message>
  </validator>
```

短路校验器

从XWork 1.0.1开始(绑定在WebWork 2.1内),短路一堆校验器变为可能.这是另外一个示例配置文件,包含了从XWork的测试用例里面的校验规则:注意一些 <field-validator> 和<validator> 元素具有 short-circuit属性并设置为true.

```
  <!DOCTYPE validators PUBLIC
          "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
          "http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
  <validators>
    <!-- Field Validators for email field -->
    <field name="email">
        <field-validator type="required" short-circuit="true">
            <message>You must enter a value for email.</message>
        </field-validator>
        <field-validator type="email" short-circuit="true">
            <message>Not a valid e-mail.</message>
        </field-validator>
    </field>
    <!-- Field Validators for email2 field -->
    <field name="email2">
       <field-validator type="required">
            <message>You must enter a value for email2.</message>
        </field-validator>
       <field-validator type="email">
            <message>Not a valid e-mail2.</message>
        </field-validator>
    </field>
    <!-- Plain Validator 1 -->
    <validator type="expression">
        <param name="expression">email.equals(email2)</param>
        <message>Email not the same as email2</message>
    </validator>
    <!-- Plain Validator 2 -->
    <validator type="expression" short-circuit="true">
        <param name="expression">email.startsWith('mark')</param>
        <message>Email does not start with mark</message>
    </validator>
  </validators>
```

短路和校验器风味

简单校验器优先字段校验器.它们首先按照它们定义的顺序被校验,然后字段校验器按照它们定义的顺序来校验.如果某个被标记为短路的校验器失败了,将会阻止其他后续的校验器的进行,然后一个错误(action错误或者字段错误,取决于校验器的类型)将会被添加到被校验的对象的ValidationContext中.

在上面的例子中,实际的校验器执行是这样的:

1. Plain Validator 1
2. Plain Validator 2
3. email字段的字段校验器
4. email2字段的字段校验器

因为普通校验器2是短路的,如果它的校验失败,它会导致email字段的校验器和email2字段的校验器不会被执行.

有用的信息: 更复杂的校验也许应该在action自己的validate()方法中进行(假定action实现了Validatable接口,ActionSupport已经实现了)

一个普通校验器(非字段校验器)如果被短路会完全打短校验过程,没有其他校验器会被执行,而普通校验器优先于字段校验器意味着普通校验器会按照它们定义的顺序被执行,会在字段校验器按照它们定义的顺序执行之前被执行.

短路校验和校验器风味

一个字段校验器如果被短路了仅会阻止其他相同字段的字段校验器被执行. Note that this "same field" behavior applies regardless of whether the or syntax was used to declare the validation rule. 例如,给出这个-validation.xml文件:

```
<validator type="required" short-circuit="true">
  <param name="fieldName">bar</param>
  <message>You must enter a value for bar.</message>
</validator>

<validator type="expression">
  <param name="expression">foo gt bar</param>
  <message>foo must be great than bar.</message>
</validator>
```

两个校验器都会执行,甚至 "required" 校验器被短路. "required" 校验器是一个字段校验器,不会短路普通校验器,因为字段校验器仅会短路相同字段上的其他校验器.因为普通校验器不是字段特定的,它不会被短路.

如何发现一个Action的校验器

正如上面所提到的,框架会搜索action的层次关系树来查找Action实现的接口和父类的缺省校验.如果你使用了短路属性,并高度依赖层次关系树中的缺省校验器, make sure you don't accidentally short-circuit things higher in the tree that you really want!

AJAX Validation

This page last changed on Apr 30, 2006 by scud.

说明

下面描述了在webwork中怎样添加一个简单的 ajax 校验.

在开始之前必须要设置了DWR servlet, 使用了dojo 和 ajax theme.

在Ajax theme 中, dwr 用来进行正常的校验,而dojo用来处理其他的事情 (widgets(页面组件), XHR, 浏览器的JS 事件(event)等等.)

为了让校验可以正常运作,建议你使用标准的WebWork Tag.

设置DWR

DWR通过位于/WEB-INF/目录下的dwr配置(dwr.xml)来设置.如果需要把它放在其他位置,请访问 http://getahead.ltd.uk/dwr/ 了解更多信息.

```
<!DOCTYPE dwr PUBLIC
        "-//GetAhead Limited//DTD Direct Web Remoting 1.0//EN"
        "http://www.getahead.ltd.uk/dwr/dwr10.dtd">

<dwr>
    <allow>
        <create creator="new" javascript="validator">
            <param name="class" value="com.opensymphony.webwork.validators.DWRValidator"/>
        </create>
        <convert converter="bean" match="com.opensymphony.xwork.ValidationAwareSupport"/>
    </allow>

    <signatures>
        <![CDATA[
        import java.util.Map;
        import com.opensymphony.webwork.validators.DWRValidator;

        DWRValidator.doPost(String, String, Map<String, String>);
        ]]>
    </signatures>
</dwr>
```

需要注册一个DWRServlet到web应用程序中.下面显示了一个典型的配置了DWRSerlvet的web.xml.

```
<servlet>
    <servlet-name>dwr</servlet-name>
    <servlet-class>uk.ltd.getahead.dwr.DWRServlet</servlet-class>
    <init-param>
        <param-name>debug</param-name>
        <param-value>true</param-value>
    </init-param>
</servlet>

<servlet-mapping>
    <servlet-name>dwr</servlet-name>
    <url-pattern>/dwr/*</url-pattern>
</servlet-mapping>
```

第一步

创建jsp页面. 注意<ww:head ...>标签用来设置theme,以放置必要的dojo脚本等等. 查看ajax的theme中的 head.ftl可以了解更多信息.

```
<html>
<head>
    <title>Validation - Basic</title>
    <ww:head theme="ajax"/>
</head>

<body>

<ww:form method="post" validate="true" theme="ajax">
    <ww:textfield label="Name" name="name"/>
    <ww:textfield label="Age" name="age"/>
    <ww:textfield label="Favorite color" name="answer"/>
    <ww:submit/>
</ww:form>

</body>
</html>
```

第二步

创建action类

```
public class QuizAction extends ActionSupport {
    String name;
    int age;
    String answer;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getAnswer() {
        return answer;
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }
}
```

第三步

创建validation.xml

```
<!--
    Add the following DOCTYPE declaration as first line of your XXX-validation.xml file:
    <!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
"http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
-->
<validators>
    <field name="name">
        <field-validator type="requiredstring">
            <message>You must enter a name</message>
        </field-validator>
    </field>
    <field name="age">
        <field-validator type="int">
            <param name="min">13</param>
            <param name="max">19</param>
            <message>Only people ages 13 to 19 may take this quiz</message>
        </field-validator>
    </field>
</validators>
```

Basic Validation

This page last changed on Apr 30, 2006 by scud.

说明

下列步骤是在Webwork中配置基本的校验.

第一步:

写一个html表单.

```
<html>
<head>
    <title>Validation - Basic</title>
    <ww:head/>
</head>

<body>

<b>What is your favorite color?</b>
<p/>

<ww:form method="post">
    <ww:textfield label="Name" name="name"/>
    <ww:textfield label="Age" name="age"/>
    <ww:textfield label="Favorite color" name="answer"/>
    <ww:submit/>
</ww:form>

</body>
</html>
```

第二步

写一个action类

```
public class QuizAction extends ActionSupport {
    String name;
    int age;
    String answer;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getAnswer() {
        return answer;
    }

    public void setAnswer(String answer) {
        this.answer = answer;
    }
}
```

第三步:

填写要使用的校验器.validation.xml的格式是 <ActionClassName>-validation.xml 或者 <ActionClassName>-<ActionContextName>-validation.xml .

```
<!--
    Add the following DOCTYPE declaration as first line of your XXX-validation.xml file:
    <!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
"http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
-->
<validators>
    <field name="name">
        <field-validator type="requiredstring">
            <message>You must enter a name</message>
        </field-validator>
    </field>
    <field name="age">
        <field-validator type="int">
            <param name="min">13</param>
            <param name="max">19</param>
            <message>Only people ages 13 to 19 may take this quiz</message>
        </field-validator>
    </field>
</validators>
```

Client Side Validation

This page last changed on May 06, 2006 by scud.

描述

WebWork在XWork的标准的校验框架的基础上增加对客户端校验的支持. 它可以基于每个表单,通过设置form标签的 validate="true" 来启用:

```
<ww:form name="test" action="javascriptValidation" validate="true">
  ...
</ww:form>
```

可以提供一个表单的 name,否则action的name就会被用作表单的name.

一个正确的 action_和 _namespace 属性必须提供给 <ww:form>标签.例如, 一个名字为 "submitProfile" 的Action,在 "/user" 命名空间(namespace)内,那么就必须使用下面的代码.

```
<ww:form namespace="/user" action="submitProfile" validate="true">
  ...
</ww:form>
```

下面的代码也会正常 "工作",表单会发挥正常的功能,但是客户端校验则不会.这是因为WebWork必须知道确切的命名空间和action(而不是一个URL)来正确地支持校验.

```
<ww:form action="/user/submitProfile.action" validate="true">
  ...
</ww:form>
```

所有的标准的 校验配置 步骤依然要进行.客户端校验使用和服务器端校验相同的校验规则.如果服务器端校验不能工作,那么客户端校验也不能工作.

注意许多WebWork 标签的 required 属性和客户端校验没有什么关系. 它只是在某个theme(例如xhtml)中用来在一个标识为必填的字段周围放置一个 '*'.

客户端校验类型

有两种形式的客户端校验:

- 纯JavaScript客户端校验 - 在 xhtml theme 和 css_xhtml theme 中使用
- AJAX 客户端校验 - 在 ajax theme 中使用

AJAX Client Side Validation

This page last changed on May 06, 2006 by scud.

基于AJAX的客户端校验是对纯JavaScript客户端校验 的改进,它使用了多种技术的组合:JavaScript,DOM操作和通过DWR进行的远端服务器通讯.不像纯客户端校验实现,基于AJAX的校验会和Server进行通讯.这意味着你的所有校验规则,如果它在提交一个表单时工作,那么它依然会和浏览器一起工作.

这种校验模式仅工作在ajax theme中

校验通过每个表单元素的 onblur 事件发生.当每一次用户输入一些内容以及移动到下一个表单元素时,输入的数据(以及和其他所有之前深入的数据)都会被发送到服务器端进行校验.整个校验stack都在进行,包括visitor 校验器和你的action的validate()方法.

如果遇到一个错误,就像纯客户端校验一样,HTML和DOM会立刻被更新.

如果想看这种校验的一个例子,请浏览 AJAX Validation.

Pure JavaScript Client Side Validation

This page last changed on May 09, 2006 by scud.

纯JavaScript客户端校验是最简单但是包含最少的客户端校验的丰富特性.这种类型的校验使用了100%的客户端JavaScript代码来设法校验用户输入的数据.因为实际上校验逻辑在JavaScript里面重复进行,所以理解这种情况很重要:有些数据会被JavaScript代码认为是可接收的,而可能会被服务器端校验认为是不可接收的.

仅有下列的校验器被支持:

required validator (必填校验器)
requiredstring validator (必填字符串校验器)
stringlength validator (字符串长度校验器)
regex validator (表达式校验器)
email validator (邮件校验器)
url validator (网址校验器)
int validator (整数校验器)
double validator (双精度数校验器)

错误报告

因为客户端校验不与服务器端通讯,xhtml theme和css\ xhtml theme负责正确地操作HTML DOM来内置方式显示错误信息.负责来进行这个逻辑的JavaScript是 validation.js 可以在每个theme里发现.

⚠ 报告错误使用的是缺省的校验信息,而不是服务器端用到的国际化版本.这是一个已知的问题.你可能会试试AJAX客户端校验 ,它的信息是国际化的.

附加的校验器支持

如果你希望在上面所列的校验器之外添加额外的校验器支持,你可能要覆盖xhtml theme的模板 form-close-validate.ftl . 这个文件包含了用来校验用户从浏览器输入的数据的JavaScript. css\ xhtml theme扩展了xhtml theme,所以它自己没有 form-close-validate.ftl 模板.

Client Validation

This page last changed on Apr 30, 2006 by scud.

说明

下面的内容是使用WebWork创建客户端校验的步骤. 注意:ww:form标签的validate属性必须设为true. 不是所有的theme都支持此特性(client-side validation)

第一步

创建jsp页面. 注意必须添加<ww:head >标签, 在这里它会设置css (xhtml theme)

```
<html>
<head>
    <title>Validation - Basic</title>
    <ww:head/>
</head>

<body>

<ww:form method="post" validate="true">
    <ww:textfield label="Name" name="name"/>
    <ww:textfield label="Age" name="age"/>
    <ww:textfield label="Favorite color" name="answer"/>
    <ww:submit/>
</ww:form>

</body>
</html>
```

第二步

创建action类

```
public class QuizAction extends ActionSupport {
    String name;
    int age;
    String answer;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public int getAge() {
        return age;
    }

    public void setAge(int age) {
        this.age = age;
    }

    public String getAnswer() {
        return answer;
    }
    public void setAnswer(String answer) {
        this.answer = answer;
    }
}
```

第三步

创建 validation.xml来配置要使用的validator(校验器).

```
<!--
    Add the following DOCTYPE declaration as first line of your XXX-validation.xml file:
    <!DOCTYPE validators PUBLIC "-//OpenSymphony Group//XWork Validator 1.0.2//EN"
"http://www.opensymphony.com/xwork/xwork-validator-1.0.2.dtd">
-->
<validators>
    <field name="name">
        <field-validator type="requiredstring">
            <message>You must enter a name</message>
        </field-validator>
    </field>
    <field name="age">
        <field-validator type="int">
            <param name="min">13</param>
            <param name="max">19</param>
            <message>Only people ages 13 to 19 may take this quiz</message>
        </field-validator>
    </field>
</validators>
```

conversion validator

This page last changed on Apr 30, 2006 by scud.

说明

字段校验器,检测是否这个字段发生过转换错误.

参数

- fieldName - 校验器要校验的字段.如果使用普通校验语法则必填,否则不必填

例子

```
<!-- Plain Validator Syntax -->
    <validator type="conversion">
                <param name="fieldName">myField</param>
         <message>Conversion Error Occurred</message>
    </validator>

    <!-- Field Validator Syntax -->
    <field name="myField">
       <field-validator type="conversion">
          <message>Conversion Error Occurred</message>
       </field-validator>
    </field>
```

译注

下面是到2006-4-5增加的内容.

发生转换错误时重新设置字段(的值)

当转换错误发生时,使用一个假的parameter map来自动设置stack,这种能力可以通过设置'repopulateField'属性为"true"来做到.

当转换错误发生时,你想要重新设置字段为原来的值,这个功能就是很有用的.例如一个textfield只允许Integer(action类声明了一个Integer字段),当转换错误发生时,不正确输入的整数(例如一个'one')当返回时不会显示出来. 通过设置'repopulateField'属性为true,它会意味着textfield发生转换错误时也会被设置为'one'.

```
<!-- myJspPage.jsp -->
<ww:form action="someAction" method="POST">
  ....
  <ww:textfield
      label="My Integer Field"
      name="myIntegerField" />
  ....
  <ww:submit />
</ww:form>
```

```
<!-- xwork.xml -->
<xwork>
<include file="webwork-default.xml" />
....
<package name="myPackage" extends="webwork-default">
  ....
  <action name="someAction" class="example.MyActionSupport.java">
     <result name="input">myJspPage.jsp</result>
     <result>success.jsp</result>
  </action>
  ....
</package>
....
</xwork>
```

```
// <!-- MyActionSupport.java -->
public class MyActionSupport extends ActionSupport {
   private Integer myIntegerField;

   public Integer getMyIntegerField() { return this.myIntegerField; }
   public void setMyIntegerField(Integer myIntegerField) {
      this.myIntegerField = myIntegerField;
   }
}
```

```
<!-- MyActionSupport-someAction-validation.xml -->
<validators>
  ...
  <field name="myIntegerField">
     <field-validator type="conversion">
        <param name="repopulateField">true</param>
        <message>Conversion Error (Integer Wanted)</message>
     </field-validator>
  </field>
  ...
</validators>
```

date validator

This page last changed on Apr 30, 2006 by scud.

说明

字段校验器,检查指定的日期是否在指定的范围内.

注意: 如果没有日期转换器被指定, 就会使用XWorkBasicConverter来进行日期转换, 缺省使用在webwork.properties里指定的或者系统默认的locale相关的Date.SHORT格式.

参数

- fieldName - 校验器校验的字段. 如果使用普通的校验语法则必填,否则不是必填
- min - 最小值. 如果没有指定, 则表示不检查.
- max - 最大值. 如果没有指定, 则表示不检查.

例子

```
<validators>
                <!-- Plain Validator syntax -->
                <validator type="date">
                <param name="fieldName">birthday</param>
          <param name="min">01/01/1990</param>
          <param name="max">01/01/2000</param>
          <message>Birthday must be within ${min} and ${max}</message>
                </validator>

      <!-- Field Validator Syntax -->
      <field name="birthday">
        <field-validator type="date">
            <param name="min">01/01/1990</param>
              <param name="max">01/01/2000</param>
              <message>Birthday must be within ${min} and ${max}</message>
        </field>
      </field>

</validators>
```

email validator

This page last changed on Apr 30, 2006 by scud.

说明

EmailValidator 检查指定的字符串字段的字符如果非空,则必须是合法的email地址.
校验字符是否合法email地址的正则表达式为:

```
\\b(^[_A-Za-z0-9-](\\.[_A-Za-z0-9-])*@([A-Za-z0-9-])+((\\.com)|(\\.net)|(\\.org)|(\\.info)|(\\.edu)|(\\.mil)
```

译注

此正则表达式可能不符合你的邮件地址的规则,请注意

参数

- fieldName - 校验器要校验的字段.如果使用普通校验语法则必填,否则不必填

例子

```
<!-- Plain Validator Syntax -->
<validators>
    <validator type="email">
        <param name="fieldName">myEmail</param>
        <message>Must provide a valid email</message>
    </validator>
</validators>

<!-- Field Validator Syntax -->
<field name="myEmail">
   <field-validator type="email">
      <message>Must provide a valid email</message>
   </field-validator>
</field>
```

expression validator

This page last changed on Apr 30, 2006 by scud.

说明

非字段级别的校验器,基于提供的正则表达式进行校验.

参数

- expression - Ognl表达式,基于stack(ValueStack)进行求值 (结果必须是Boolean值)

例子

```
<validators>
      <validator type="expression">
         <param name="expression"> .... </param>
         <message>Failed to meet Ognl Expression  .... </message>
      </validator>
</validators>
```

fieldexpression validator

This page last changed on Apr 30, 2006 by scud.

说明

使用OGNL表达式校验字段.

参数

- fieldName - 校验器校验的字段名. 使用普通的校验语法则必填,否则不是必填
- expression - Ognl表达式,基于stack进行求值 (结果必须是Boolean值)

例子

```
<!-- Plain Validator Syntax -->
<validators>
    <!-- Plain Validator Syntax -->
    <validator type="fieldexpression">
       <param name="fieldName">myField</param>
       <param name="expression"><![CDATA[#myCreditLimit > #myGirfriendCreditLimit]]></param>
       <message>My credit limit should be MORE than my girlfriend</message>
    <validator>

    <!-- Field Validator Syntax -->
    <field name="myField">
        <field-validator type="fieldexpression">
            <param name="expression"><![CDATA[#myCreditLimit >
#myGirfriendCreditLimit]]></param>
            <message>My credit limit should be MORE than my girlfriend</message>
        </field-validator>
    </field>

</vaidators>
```

int validator

This page last changed on Apr 30, 2006 by scud.

说明

int validator 检查指定的整数是否在一定范围内.

参数

- fieldName - 校验器校验的字段名.如果使用普通校验语法则必填,否则不需要
- min - 最小值 (如果没有指定, 则不检查)
- max - 最大值 (如果没有指定, 则不检查)

例子

```
<validators>
          <!-- Plain Validator Syntax -->
          <validator type="int">
              <param name="fieldName">age</param>
              <param name="min">20</param>
              <param name="max">50</param>
              <message>Age needs to be between ${min} and ${max}</message>
          </validator>

          <!-- Field Validator Syntax -->
          <field name="age">
              <field-validator type="int">
                  <param name="min">20</param>
                  <param name="max">50</param>
                  <message>Age needs to be between ${min} and ${max}</message>
              </field-validator>
          </field>
     </validators>
```

regex validator

This page last changed on Apr 30, 2006 by scud.

说明

使用正则表达式校验一个字符串字段

参数

- fieldName - 校验器要校验的字段.如果使用普通校验语法则必填,否则不必填
- expression - 正则表达式,必须的属性
- caseSensitive - 布尔值 (可选). 设置表达式匹配时是否大小写敏感.缺省为true.(译注:此属性为后来源码里加上的)

例子

```
<validators>
    <!-- Plain Validator Syntax -->
    <validator type="regex">
        <param name="fieldName">myStrangePostcode</param>
        <param name="expression"><![CDATA[([aAbBcCdD][123][eEfFgG][456])]]<>/param>
    </validator>

    <!-- Field Validator Syntax -->
    <field name="myStrangePostcode">
        <field-validator type="regex">
           <param name="expression"><![CDATA[([aAbBcCdD][123][eEfFgG][456])]]></param>
        </field-validator>
    </field>
</validators>
```

required validator

This page last changed on Apr 05, 2006 by dogg.

说明

RequiredFieldValidator 校验器检查指定的字段必须有值(非空).

参数

- fieldName - 字段名
  如果不使用简单校验语法,此参数不需要定义.

例子

```
<validators>

        <!-- ####### -->
        <validator type="required">
            <param name="fieldName">username</param>
            <message>username must not be null</message>
        </validator>


        <!-- Field Validator Syntax -->
        <field name="username">
            <field-validator type="required">
                   <message>username must not be null</message>
            </field-validator>
        </field>

    </validators>
```

requiredstring validator

This page last changed on Apr 30, 2006 by scud.

说明

RequiredStringValidator 校验器检查字段值必须非空且长度 > 0. (也就是不能是 ""). "trim" 参数决定是否在检测之前是否trim,如果没有指定,字符串将会被trim.

参数

- fieldName - 校验器校验的字段名.如果使用普通校验语法则必须填写,否则不需要
- trim - 校验之前trim字段的值 (缺省为 true)

例子

```
<validators>
    <!-- Plain-Validator Syntax -->
    <validator type="requiredstring">
        <param name="fieldName">username</param>
        <param name="trim">true</param>
        <message>username is required</message>
    </validator>

    <!-- Field-Validator Syntax -->
    <field name="username">
          <field-validator type="requiredstring">
            <param name="trim">true</param>
            <message>username is required</message>
       </field-validator>
    </field>
</validators>
```

stringlength validator

This page last changed on Apr 30, 2006 by scud.

说明

StringLengthFieldValidator检查一个字符串字段是否在一定的长度范围内. 如果"minLength" 参数指定了,它会确保字符串拥有那么多字符.如果 "maxLength" 参数指定了,它会确保字符串最多有那么多字符. "trim" 参数决定在检查长度之前是否进行trim. 如果没有指定,字符串会被trim.

参数

fieldName - 校验器校验的字段名.如果使用普通校验语法则必填,否则不需要
maxLength - 字段值的最大长度. 缺省不检查.
minLength - 字段值的最小长度. 缺省不检查.
trim - 检测min/max长度之前是否trim字段值.缺省为 true

例子

```
<validators>
         <!-- Plain Validator Syntax -->
                        <validator type="stringlength">
                                <param name="fieldName">myPurchaseCode</param>
                                <param name="minLength">10</param>
             <param name="maxLength">10</param>
             <param name="trim">true</param>
             <message>Your purchase code needs to be 10 characters long</message>
         </validator>

                        <!-- Field Validator Syntax -->
                        <field name="myPurchaseCode">
                                <param name="minLength">10</param>
             <param name="maxLength>10</param>
             <param name="trim">true</param>
             <message>Your purchase code needs to be 10 characters long</message>
                        </field-name>
     </validators>
```

url validator

This page last changed on Apr 30, 2006 by scud.

说明

URLValidator 检查指定的字段是否是字符串并且是合法的URL

参数

- fieldName - 校验器要校验的字段.如果使用普通校验语法则必填,否则不必填

例子

```
<validators>
         <!-- Plain Validator Syntax -->
         <validator type="url">
             <param name="fieldName">myHomePage</param>
             <message>Invalid homepage url</message>
         </validator>

         <!-- Field Validator Syntax -->
         <field name="myHomepage">
             <message>Invalid homepage url</message>
         </field>
    </validators>
```

Using Field Validators

This page last changed on Apr 30, 2006 by scud.

说明

下面展示了在Webwork中使用字段校验器的一个简单例子.

第一步

创建jsp页面

```
<h3>All Field Errors Will Appear Here</h3>
<ww:fielderror />
<hr/>

<h3>Field Error due to 'Required String Validator Field' Will Appear Here</h3>
<ww:fielderror>
        <ww:param value="%{'requiredStringValidatorField'}" />
</ww:fielderror>
<hr/>

<h3>Field Error due to 'String Length Validator Field' Will Appear Here</h3>
<ww:fielderror>
        <ww:param>stringLengthValidatorField</ww:param>
</ww:fielderror>
<hr/>

<ww:form action="submitFieldValidatorsExamples" namespace="/validation" method="POST"
theme="xhtml">
        <ww:textfield label="Required Validator Field" name="requiredValidatorField" />
        <ww:textfield label="Required String Validator Field"
name="requiredStringValidatorField" />
        <ww:textfield label="Integer Validator Field" name="integerValidatorField" />
        <ww:textfield label="Date Validator Field" name="dateValidatorField" />
        <ww:textfield label="Email Validator Field" name="emailValidatorField" />
        <ww:textfield label="URL Validator Field" name="urlValidatorField" />
        <ww:textfield label="String Length Validator Field" name="stringLengthValidatorField"
/>
        <ww:textfield label="Regex Validator Field" name="regexValidatorField"/>
        <ww:textfield label="Field Expression Validator Field"
name="fieldExpressionValidatorField" />
        <ww:submit label="Submit" />
</ww:form>
```

第二步

创建actoin类. 译者注:AbstractValidationActionSupport类在WebWork发行包的webapps\showcase\src\java\com\opensymphony\webwork\showcase\validation下

```
public class FieldValidatorsExampleAction extends AbstractValidationActionSupport {

        private String requiredValidatorField = null;
        private String requiredStringValidatorField = null;
        private Integer integerValidatorField = null;
        private Date dateValidatorField = null;
        private String emailValidatorField = null;
        private String urlValidatorField = null;
        private String stringLengthValidatorField = null;
        private String regexValidatorField = null;
        private String fieldExpressionValidatorField = null;


        private String randomNumber = null;


        public Date getDateValidatorField() {
                return dateValidatorField;
        }
        public void setDateValidatorField(Date dateValidatorField) {
                this.dateValidatorField = dateValidatorField;
        }
        public String getEmailValidatorField() {
                return emailValidatorField;
        }
        public void setEmailValidatorField(String emailValidatorField) {
                this.emailValidatorField = emailValidatorField;
        }
        public Integer getIntegerValidatorField() {
                return integerValidatorField;
        }
        public void setIntegerValidatorField(Integer integerValidatorField) {
                this.integerValidatorField = integerValidatorField;
        }
        public String getRegexValidatorField() {
                return regexValidatorField;
        }
        public void setRegexValidatorField(String regexValidatorField) {
                this.regexValidatorField = regexValidatorField;
        }
        public String getRequiredStringValidatorField() {
                return requiredStringValidatorField;
        }
        public void setRequiredStringValidatorField(String requiredStringValidatorField) {
                this.requiredStringValidatorField = requiredStringValidatorField;
        }
        public String getRequiredValidatorField() {
                return requiredValidatorField;
        }
        public void setRequiredValidatorField(String requiredValidatorField) {
                this.requiredValidatorField = requiredValidatorField;
        }
        public String getStringLengthValidatorField() {
                return stringLengthValidatorField;
        }
        public void setStringLengthValidatorField(String stringLengthValidatorField) {
                this.stringLengthValidatorField = stringLengthValidatorField;
        }
        public String getFieldExpressionValidatorField() {
                return fieldExpressionValidatorField;
        }
        public void setFieldExpressionValidatorField(
                        String fieldExpressionValidatorField) {
                this.fieldExpressionValidatorField = fieldExpressionValidatorField;
        }

    public String getUrlValidatorField() {
        return urlValidatorField;
    }

    public void setUrlValidatorField(String urlValidatorField) {
        this.urlValidatorField = urlValidatorField;
    }
}
```

第三步

创建validator.xml.

```
<validators>
        <field name="requiredValidatorField">
```

```
                <field-validator type="required">
                        <message><![CDATA[ required ]]></message>
                </field-validator>
        </field>
        <field name="requiredStringValidatorField">
                <field-validator type="requiredstring">
                        <param name="trim">true</param>
                        <message><![CDATA[ required and must be string ]]></message>
                </field-validator>
        </field>
        <field name="integerValidatorField">
                <field-validator type="int">
                        <param name="min">1</param>
                        <param name="max">10</param>
                        <message><![CDATA[ must be integer min 1 max 10 if supplied
]]></message>
                </field-validator>
        </field>
        <field name="dateValidatorField">
                <field-validator type="date">
                        <param name="min">01/01/1990</param>
                        <param name="max">01/01/2000</param>
                        <message><![CDATA[ must be a min 01-01-1990 max 01-01-2000 if supplied
]]></message>
                </field-validator>
        </field>
        <field name="emailValidatorField">
                <field-validator type="email">
                        <message><![CDATA[ must be a valid email if supplied ]]></message>
                </field-validator>
        </field>
        <field name="urlValidatorField">
                <field-validator type="url">
                        <message><![CDATA[ must be a valid url if supplied ]]></message>
                </field-validator>
        </field>
        <field name="stringLengthValidatorField">
                <field-validator type="stringlength">
                        <param name="maxLength">4</param>
                        <param name="minLength">2</param>
                        <param name="trim">true</param>
                        <message><![CDATA[ must be a String of a specific greater than 1 less
than 5 if specified ]]></message>
                </field-validator>
        </field>
        <field name="regexValidatorField">
                <field-validator type="regex">
                        <param name="expression">.*\.txt</param>
                        <message><![CDATA[ regexValidatorField must match a regexp (.*\.txt) if
specified ]]></message>
                </field-validator>
        </field>
        <field name="fieldExpressionValidatorField">
                <field-validator type="fieldexpression">
                        <param name="expression">(fieldExpressionValidatorField ==
requiredValidatorField)</param>
                        <message><![CDATA[ must be the same as the Required Validator Field if
specified ]]></message>
                </field-validator>
        </field>
</validators>
```

Using Non Field Validators

This page last changed on Apr 30, 2006 by scud.

说明

下面展示了在Webwork中使用非字段校验器(Non Field Validators)的一个简单例子.

第一步

创建jsp页面

```
<ww:actionerror />

<ww:form method="POST" action="submitNonFieldValidatorsExamples" namespace="/validation">
        <ww:textfield name="someText" label="Some Text" />
        <ww:textfield name="someTextRetype" label="Retype Some Text" />
        <ww:textfield name="someTextRetypeAgain" label="Retype Some Text Again" />
        <ww:submit label="Submit" />
</ww:form>
```

第二步

创建action类

```
public class NonFieldValidatorsExampleAction extends AbstractValidationActionSupport {

        private String someText;
        private String someTextRetype;
        private String someTextRetypeAgain;

        public String getSomeText() {
                return someText;
        }
        public void setSomeText(String someText) {
                this.someText = someText;
        }
        public String getSomeTextRetype() {
                return someTextRetype;
        }
        public void setSomeTextRetype(String someTextRetype) {
                this.someTextRetype = someTextRetype;
        }
        public String getSomeTextRetypeAgain() {
                return someTextRetypeAgain;
        }
        public void setSomeTextRetypeAgain(String someTextRetypeAgain) {
                this.someTextRetypeAgain = someTextRetypeAgain;
        }
}
```

第三步

创建validator.xml.

```
<validators>
        <validator type="expression">
                <param name="expression"><![CDATA[ ( (someText == someTextRetype) &&
(someTextRetype == someTextRetypeAgain) ) ]]></param>
                <message><![CDATA[ all three text must be exactly the same ]]></message>
        </validator>
</validators>
```

Using Visitor Field Validator

This page last changed on Apr 30, 2006 by scud.

说明

这里演示了一个使用WebWork的Visitor字段校验器的简单例子.

第一步

创建jsp页面

```
<ww:fielderror />

<ww:form method="POST" action="submitVisitorValidatorsExamples" namespace="/validation">
        <ww:textfield name="user.name" label="User Name" />
        <ww:textfield name="user.age" label="User Age" />
        <ww:textfield name="user.birthday" label="Birthday" />
        <ww:submit label="Submit" />
</ww:form>
```

第二步

创建action类

```
public class VisitorValidatorsExampleAction extends AbstractValidationActionSupport {

        private User user;

        public User getUser() {
                return user;
        }

        public void setUser(User user) {
                this.user = user;
        }
}
```

第三步

创建 validator.xml.

```
<validators>
        <field name="user">
                <field-validator type="visitor">
                        <param name="context">userContext</param>
                        <param name="appendPrefix">true</param>
                        <message>User:</message>
                </field-validator>
        </field>
</validators>
```

visitor validator

This page last changed on Apr 30, 2006 by scud.

说明

VisitorFieldValidator 允许转换校验到(使用对象自己的校验文件)校验你的action里的对象的属性.着允许你使用模型驱动(ModelDriver)开发模式并在一个地方管理你的model的校验,也就是你的model类所在的地方.VisitorFieldValidator 可以处理简单的对象属性,也可以处理对象集合或者对象数组.

参数

- fieldName - 如果普通的校验语法使用,则设为字段名.如果字段校验语法使用则不需要
- context - 校验会发生的context. 可选
- appendPrefix - 要添加到字段上的前缀.可选

例子

```
<validators>
    <!-- Plain Validator Syntax -->
    <validator type="visitor">
        <param name="fieldName">user</param>
        <param name="context">myContext</param>
        <param name="appendPrefix">true</param>
    </validator>

    <!-- Field Validator Syntax -->
    <field name="user">
       <field-validator type="visitor">
          <param name="context">myContext</param>
          <param name="appendPrefix">true</param>
       </field-validator>
    </field>
</validators>
```

在上面的例子中,如果action的getUser()方法返回一个User对象,WebWork为了校验会查找User-myContext-validation.xml.因为appendPrefix为true,每个字段名将会被加上'user'前缀,例如如果实际的字段名是'name',那么将得到'user.name'.

Velocity

This page last changed on Apr 10, 2006 by scud.

Velocity是一种Java模版语言.

关于Velocity本身的更多信息, 请访问Velocity 网站.

Velocity与FreeMarker非常相似, 它们都是可以在Servlet容器外使用的模版语言. WebWork团队更加推荐使用FreeMarker, 因为它有比Velocity更好的错误报告, 支持JSP tags, 还有其它一些细微优点. 当然, 它们都是JSP很好的替代品.

上手

开始使用Velocity, 首先需要保证所有的依赖都已经被加入到你的工程的classpath. 其次, webwork\-default.xml 中要配置好Velocity Result, 它将影射你的action和模版. 你可以使用如下的 `xwork.xml` 配置:

```
<action name="test" class="com.acme.TestAction">
    <result name="success" type="velocity">test-success.vm</result>
</action>
```

然后在 `test-success.vm` 中:

```
<html>
<head>
    <title>Hello</title>
</head>
<body>

Hello, ${name}

</body>
</html>
```

Where `name` is a property on your action. That's it! Read the rest of this document for details on how templates are loaded, variables are resolved, and tags can be used.
这里的 `name` 是你的action中的一个属性. 这就行了! 阅读余下的文档你将了解模版的载入, 变量如何解析, 和tags的使用.

模版载入

WebWork looks for Velocity templates in two locations (in this order):
WebWork在两个位置寻找Velocity模版(按顺序):

1. Web应用程序
2. Class path

这个顺序使在编译好的jar包中提供模版非常方便, 同时还允许你在你的web应用程序中覆盖这些模版. 实际上, 这就是你可以覆盖WebWork默认的UI tags 和 Form Tags模版的原因.

变量解析

在Velocity中，将从几个不同的位置查找变量，顺序如下：

1. 值栈(value stack)
2. action上下文(context)
3. 内建变量

注意action上下文在值栈后面. 着一位着你可以不用加上典型的前缀 (#) 引用变量，就像你在使用JSP ww:property 标签时一样. 这很方便，但是请小心，有时它也有可能会给你带来小麻烦.

```
#wwurl "id=url "value=http://www.yahoo.com"
Click <a href="${url}">here</a>!
```

The built-in variables that WebWork-Velocity integration provides are:
WebWork-Velocity集成中内建的变量是:

名称	描述
stack	值栈(value stack)本身，使用这样的语法访问 ${stack.findString('ognl expr')}
action	最近执行的action
response	HttpServletResponse
res	与response相同
request	HttpServletRequest
req	与request相同
session	HttpSession
application	ServletContext
base	request的上下文路径(context path)

Tag支持

See the Velocity Tags documentation for information on how to use the generic Tags provided by WebWork.
察看Velocity标签文档获取更多关于如何使用WebWork提供的一般标签的内容.

提示与技巧

There are some advanced features that may be useful when building WebWork applications with Velocity.
当在WebWork应用程序中使用Velocity时，有一些进阶的功能可能会游泳.

扩展

有时你可能希望扩展WebWork提供的Velocity支持. 最常见的原因是你希望加入你自己的标签，就像你从WebWork内建的标签中扩展一样.

想要这样做, 写一个新的扩展自 com.opensymphony.webwork.views.velocity.VelocityManager 的类或者也可以覆盖它.
然后增加如下内容到webwork.properties:

```
webwork.velocity.manager.classname = com.yourcompany.YourVelocityManager
```

配置

你可以通过替换velocity.properties中的配置项来配置Velocity.

velocity.properties

This page last changed on Feb 02, 2006 by zjumty.

如果您提供了这个文件(在/WEB-INF/classes下),Velocity就会加载它.它可以用来加载自定义宏:

```
# Velocity Macro libraries.
velocimacro.library = webwork.vm, tigris-macros.vm, myapp.vm
```

更多信息参见Velocity文档

XWork Configuration

This page last changed on Feb 15, 2006 by scud.

概况

所有Action的配置在xwork.xml中已经有所描述(更多信息查看配置文件).这一部分主要介绍组成Action配置的各个元素,例如actions,interceptors,results和package.

Action configuration

This page last changed on May 18, 2006 by scud.

Action是Webwork的基础"工作单元". 一个action一般就是一个请求(或点击按钮或提交表单). action元素(tag就和JSP太同义了)有两部分, 一个友好的名字(URL相关, 如saveForm. action)和一个负责"处理"的类.

```
<action name="formTest" class="com.opensymphony.webwork.example.FormAction"
method="processForm">
```

可选属性"**method**"用来告诉WebWork调用action的那个方法. 如果method属性为空, WebWork调用默认调用*execute*()方法. 如果Action类中既没有execute()方法也没有在xml文件中指定其他方法, WebWork会抛出异常.

您也可以在您的表单中用"**actionName!something**"的方式告诉WebWork调用Action类中的"**something**"方法. 例如"**formTest!save.action**"会调用FormAction类中的"**save**"方法. 这个方法必须是public且没有参数.

```
public String save() throws Exception{
...
return SUCCESS;
}
```

"**actionName**"的所有配置包括拦截器和返回类型等都会被"actionName!something"使用.

Action Support

Action的类属性可以像下面一样省略:

```
<action name="myAction">
   ....
</action>
```

在这种情况下, 会缺省使用com.opensymphony.xwork.ActionSupport 类, 它有一个execute()方法, 缺省返回SUCCESS.

默认Action引用

从Webwork2.2.1开始您也可以指定一个当xwork.xml中找不到指定的action时执行的默认action. 这一特性主要是用来满足为创建非常简单或相似的action类或元素的需求. 默认action名可以在package元素里面这样配置:

```
<package name="myPackage" ....>

...

<default-action-ref name="simpleViewResultAction">

<!--
An example of a default action that is just a simple class
that has 3 fields: successUrl, errorUrl, and inputUrl. This action
parses the request url to set the result values. In the normal case
it just renders velocity results of the same name as the requested url.
-->
<action name="simpleViewResultAction" class="SimpleViewResultAction">
<result type="velocity">${successUrl}</result>
<result name="error" type="velocity">${errorUrl}</result>
```

```
    <result name="input" type="velocity">${inputUrl}</result>
    </action>

    ...

    </package>
```

当心

应该保证在每个名称空间中只有一个默认action被配置.因此如果你在同一个空间里配置了多个默认aciton,系统就不知道那个是默认的了.

注意

注意第一个result的属性省略了,WebWork缺省会把它当作"success".

在这种情况下,如果请求到action的映射没有在这个包里定义,它会自动转向到别名为 "simpleViewResultAction" 的action来执行.

这里的大多数内容都是 Matt Dowell <matt.dowell@notiva.com> 提供的.

Include configuration

This page last changed on May 18, 2006 by scud.

介绍

为了方便的管理大型项目(非常多的action和配置),Webwork允许您在xwork.xml中包含其他的配置文件.

```
<xwork>
    <include file="webwork-default.xml"/>
    <include file="user.xml"/>
    <include file="shoppingcart.xml"/>
    <include file="product.xml"/>
    ....
</xwork>
```

被包含的文件必须是xwork.xml的格式(有doctype和其他每样东西),而且必须放在classpath下(通常是/WEB-INF/classes或/WEB-INF/lib下的jar文件中).

这里的大多内容由Matt Dowell提供<matt.dowell@notiva.com>

Interceptor Configuration

This page last changed on May 19, 2006 by scud.

描述

拦截器允许您定义一些能在一个action执行的前后执行的代码.它是做web应用程序时很有用的工具.最常见的拦截器实现可以是:

- 安全检查(确保访问者是登陆用户)
- 跟踪日志(记录每个action)
- 效率瓶颈检查(记录每个action开始和结束的时间以检查程序中的瓶颈)

您也可以把拦截器连在一起组成 **拦截器栈**.如果您想在action执行前同时做登陆检查,安全检查和记录日志,可以定义一个拦截器的包.

拦截器必须事先定义(命名)好,然后可以连在一起组成一个栈.

```
<interceptors>
  <interceptor name="security" class="com.mycompany.security.SecurityInterceptor"/>
  <interceptor-stack name="defaultComponentStack">
    <interceptor-ref name="component"/>
    <interceptor-ref name="defaultStack"/>
  </interceptor-stack>
</interceptors>
```

在action中这样使用.

```
<action name="VelocityCounter" class="com.opensymphony.webwork.example.counter.SimpleCounter">
   <result name="success">...</result>
   <interceptor-ref name="defaultComponentStack"/>
</action>
```

注意:引用名既可以是拦截器名也可以是栈名

更多信息参见拦截器章节的介绍

这里的内容大部分是 Matt Dowell <matt.dowell@notiva.com> 提供的

Namespace Configuration

This page last changed on May 18, 2006 by scud.

名称空间

名称空间属性允许把action配置分成不同的名称空间,这样您可以在具有不同类和参数名称空间中使用相同的名字的action了.这一点是和Webwork1.x不同,在Webwork1.x中所有的action名字都是全局的不能再一个应用程序中重用.

默认名称空间

默认名称空间用""(空字符串)表示.如果系统在指定的名称空间中没有找到某个action,就会到默认名称空间中查找.你可以在所有用"extends"扩展的名称空间外配置全局action,就像Webwork 1.x那样不指定名称空间.名称空间也可以用来实现系统安全,例如action名字之前可以有个路径,这个路径就是Webwork 2.0 ServletDispatcher用的名称空间,你可以在路径上使用J2EE声明式的安全限制，这种方式很容易的实现和维护。

根名称空间

Webwork中有以"/"命名的根名称空间,它是请求直接来自应用程序根路径的时候的名称空间.和其他名称空间一样,如果在根名称空间中没有所需的action别名,系统会回到默认名称空间中查找.

名称空间的例子

```
<package name="default">
    <action name="foo" class="mypackage.simpleAction>
        <result name="success" type="dispatcher">greeting.jsp</result>
    </action>
    <action name="bar" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">bar1.jsp</result>
    </action>
</package>

<package name="mypackage1" namespace="/">
    <action name="moo" class="mypackage.simpleActtion">
        <result name="success" type="dispatcher">moo.jsp</result>
    </action>
</package>

<package name="mypackage2" namespace="/barspace">
    <action name="bar" class="mypackage.simpleAction">
        <result name="success" type="dispatcher">bar2.jsp</result>
    </action>
</package>
```

解释

如果请求为/barspace/bar.action,系统首先查找'/barspace'名称空间,如果找到bar action便执行,如果没有继续到默认名称空间中查找.在这个例子中'/barspace'空间中存在bar别名,所以它会被执行,如果返回success结果,请求将指向bar2.jsp.

注意:如果请求为/barspace/foo.action,系统会在/barspace空间中查找foo这个action,如果找不到,系统继续在默认名称空间中查找.除非指定成其他的,否则默认空间是"".在我们上面的例子中,/barspace空间中没有foo这个action,这样默认空间中的/foo.action会被找到并执行.

如果请求为/moo.action,系统会在根空间('/')中查找'moo' action别名,如果没找到再到默认空间中查找.在这个例子中,moo这个action别名存在因此会被执行.如果返回success,请求指向bar2.jsp.

注意:如果请求为'/foo.action',系统会在'/'空间中查找,找到后执行,如果没有,继续查找默认空间.在本例中,foo这个action别名不存在于'/'空间,所以系统回到默认空间中查找并执行.

注意:名称空间只有一个级别.例如如果url是'/barspace/myspace/bar.action',Webwork先试着查找'/barspace/myspace',在本例中是不存在的.接着就直接到''空间中查找'bar'这个action别名.结果在默认空间中的bar会被执行.

Package Configuration

This page last changed on May 18, 2006 by scud.

概况

Packages是把Actions, Results, Result Types, Interceptors和Stacks分组成逻辑单元的一种方式,以分享一些共同设置.包跟对象一样可以扩展,也可以被"子"包覆盖部分属性.

包

"name"属性是package的必需,在这个package被引用时作为key. "extends"属性是可选的,使一个package可以继承一个或多个前面package中的拦截器,拦截器栈,action等配置.注意,配置文件是被至上而下处理的,所以"被继承"的包必需定义在"继承"包的上面. "abstract"是可选的表示一个包为抽象包,这样的包可以被继承,但是不能有action的定义.

属性	必需	描述
name	是	被其它包引用时的key
extends	否	继承其它包
namespace	否	参见名称空间的配置
abstract	否	把包声明为抽象(包中没有action定义)

xwork.xml中包的例子

```
<xwork>

    <include file="webwork-default.xml"/>

    <include file="config-browser.xml"/>

    <include file="xwork-continuations.xml"/>

    <include file="xwork-tags.xml"/>

    <include file="xwork-validation.xml" />

    <include file="xwork-actionchaining.xml" />

    <include file="xwork-ajax.xml" />

    <include file="xwork-fileupload.xml" />

    <include file="xwork-person.xml" />

    <include file="xwork-wait.xml" />

    <include file="xwork-token.xml" />

    <include file="xwork-model-driven.xml" />

    <include file="xwork-filedownload.xml" />

    <package name="default" extends="webwork-default">
        <interceptors>
            <interceptor-stack name="crudStack">
                <interceptor-ref name="params" />
                <interceptor-ref name="defaultStack" />
            </interceptor-stack>
        </interceptors>

                <default-action-ref name="showcase"/>

        <action name="showcase">
            <result>showcase.jsp</result>
        </action>

        <action name="date" class="com.opensymphony.webwork.showcase.DateAction">
            <result name="success">/date.jsp</result>
        </action>

    </package>

    <package name="skill" extends="default" namespace="/skill">
        <default-interceptor-ref name="crudStack"/>

        <action name="list" class="com.opensymphony.webwork.showcase.action.SkillAction"
method="list">
            <result>/empmanager/listSkills.jsp</result>
            <interceptor-ref name="basicStack"/>
        </action>
        <action name="edit" class="com.opensymphony.webwork.showcase.action.SkillAction">
            <result>/empmanager/editSkill.jsp</result>
            <interceptor-ref name="params" />
            <interceptor-ref name="basicStack"/>
        </action>
        <action name="save" class="com.opensymphony.webwork.showcase.action.SkillAction"
method="save">
            <result name="input">/empmanager/editSkill.jsp</result>
            <result type="redirect">edit.action?skillName=${currentSkill.name}</result>
        </action>
        <action name="delete" class="com.opensymphony.webwork.showcase.action.SkillAction"
method="delete">
            <result name="error">/empmanager/editSkill.jsp</result>
            <result type="redirect">edit.action?skillName=${currentSkill.name}</result>
        </action>
    </package>

    <package name="employee" extends="default" namespace="/employee">
        <default-interceptor-ref name="crudStack"/>

        <action name="list" class="com.opensymphony.webwork.showcase.action.EmployeeAction"
method="list">
            <result>/empmanager/listEmployees.jsp</result>
            <interceptor-ref name="basicStack"/>
        </action>
        <action name="edit" class="com.opensymphony.webwork.showcase.action.EmployeeAction">
            <result>/empmanager/editEmployee.jsp</result>
            <interceptor-ref name="crudStack"><param
name="validation.excludeMethods">execute</param></interceptor-ref>
        </action>
        <action name="save" class="com.opensymphony.webwork.showcase.action.EmployeeAction"
method="save">
            <result name="input">/empmanager/editEmployee.jsp</result>
            <result type="redirect">edit.action?empId=${currentEmployee.empId}</result>
        </action>
        <action name="delete" class="com.opensymphony.webwork.showcase.action.EmployeeAction"
method="delete">
            <result name="error">/empmanager/editEmployee.jsp</result>
            <result type="redirect">edit.action?empId=${currentEmployee.empId}</result>
        </action>
    </package>

</xwork>
```

Result Configuration

This page last changed on May 18, 2006 by scud.

描述

Result是Action返回的表示Action执行情况的字符串常量.WebWork定义了一些默认结果:error, input, login, none and success.开发者当然也可以根据应用情况自由的定义结果.结果以"名字-值"的形式影射到结果类型.

- 全局结果
- 默认结果

结果标签

结果标签告诉WebWork在action被调用以后下一步做什么.这里是WebWork定义好的一些结果编码:

```
String SUCCESS = "success";
String NONE    = "none";
String ERROR   = "error";
String INPUT   = "input";
String LOGIN   = "login";
```

您可以自己扩展.大多数情况下你会用到 SUCCESS 或 ERROR ,当返回 SUCCESS 时跳转到下应用程序的一个页面.

```
<result name="success" type="dispatcher">
    <param name="location">/thank_you.jsp</param>
</result>
```

...如果返回*ERROR*转向错误页面或回到前面的页面.

```
<result name="error" type="dispatcher">
    <param name="location">/error.jsp</param>
</result>
```

结果在xwork.xml文件中定义,嵌套在<action>标签里.如果location参数是唯一的参数，你可以这样简单的定义:

```
<action name="bar" class="myPackage.barAction">
  <result name="success" type="dispatcher">
    <param name="location">foo.jsp</param>
  </result>
</action>
```

或简单的

```
<action name="bar" class="myPackage.barAction">
  <result name="success" type="dispatcher">foo.jsp</result>
</action>
```

甚至更简单

```
<action name="bar" class="myPackage.barAction">
   <result>foo.jsp</result>
</action>
```

默认Action类

如果action标签中的class属性没有指定,系统默认为WebWork的ActionSupport类.

默认Location参数

如果<result ..>标签中没有param标签,如<param name="location"> ,,, </param>作为子标签.WebWork就把<result> </result>里面的文字作为location

默认返回类型

如果没有指定<result ...>标签的type属性,WebWork默认为dispatcher类型(类似于Servlet标准中的SerlvetDispatcher的forward)

Default results

This page last changed on May 18, 2006 by scud.

描述

在WebWork中您可以为您的Action定义默认的结果类型.这样当使用默认结果类型时就不用指定了.如果一个包扩展另一个包,且您没有为子包指定新的默认结果类型,那么当子包的result标签中没有指定结果类型时就会使用父包中的默认类型.

```
<!-- parts of xwork.xml  -->
....

<result-types>
 <result-type name="dispatcher"
class="com.opensymphony.webwork.dispatcher.ServletDispatcherResult" default="true"/>
 <result-type name="redirect"
class="com.opensymphony.webwork.dispatcher.ServletRedirectResult"/>
 <result-type name="velocity" class="com.opensymphony.webwork.dispatcher.VelocityResult"/>
</result-types>

....

<action name="bar" class="myPackage.barAction">

<!-- this result uses dispatcher, so you can omit the type="dispatcher" if you want -->
  <result name="success">foo.jsp</result>

<!-- this result uses velocity result, so the type needs to be specified -->
  <result name="error" type="velocity">error.vm</result>

</action>

....
```

Global results

This page last changed on Feb 18, 2006 by scud.

描述

WebWork允许您为所有Action配置定义一些默认的result映射,它会自动的被这个包中所有的Action以及所有扩展包继承,换句话说,如果您在多个Action中使用相同的result映射,您可以把它配置成全局的Result.

例子

```
<package name="default">
....
<global-results>
    <result name="login" type="dispatcher">
        <param name="location">login.jsp</param>
    </result>
</global-results>
<action name="foo"  class="mypackage.fooAction">
    <result name="success" type="dispatcher">bar.jsp</result>
</action>
<action name="submitForm"  class="mypackage.submitFormAction">
    <result name="success" type="dispatcher">submitSuccess.jsp</result>
</action>
...
</package>
```

这样配置也可以

```
<package name="default">
....
<action name="foo"  class="mypackage.fooAction">
    <result name="success" type="dispatcher">bar.jsp</result>
    <result name="login" type="dispatcher">login.jsp</result>
</action>
<action name="submitForm"  class="mypackage.submitFormAction">
    <result name="success" type="dispatcher">submitSuccess.jsp</result>
    <result name="login" type="dispatcher">login.jsp</result>
</action>
...
</package>
```

Views

This page last changed on May 18, 2006 by scud.

描述

WebWork支持JSP和Velocity(译者注:还有Freemaker,Xslt,Groovy等)作为表现层技术.这个例子中我们使用JSP文件.WebWork提供了一套标签库(taglibs).你可以在JSP文件中把这些标签作为组件使用.这里是form.jsp页的一部分.

```
<%@ taglib prefix="ww" uri="webwork" %>
<html>
<head><title>Webwork Form Example</title></head>
<body>
   <ww:form name="myForm" action="'formTest'" namespace="/" method="POST">
  <table>
    <ww:textfield label="First Name" name="'formBean.firstName'" value="formBean.firstName"/>
    <ww:textfield label="Last Name" name="'formBean.lastName'" value="formBean.lastName"/>
    <ww:submit value="Save Form"/>
  </table>
</ww:form>
</body>
```

(译者注:上面这段代码中的语法使用的是老的标签语法,如果使用alt语法则略有不同)

处理流程如下:

1. WebWork会对以 `.action` 结尾的URI进行处理(定义在 `web.xml` 文件中)(译者注:URI的模式是可以替换的)
2. WebWork查找名为 `formTest` 的action,调用定义在其上的拦截器.
3. WebWork翻译 `formTest` 然后决定调用定义在xwork.xml文件中的 `com.opensymphony.webwork.example.FormAction` 类中的 `processForm` 方法.
4. 这个方法执行成功后返回 `SUCCESS` 作为结果
5. WebWork按照xwork.xml中的定义把 `SUCCESS` 转义为 `formSuccess.jsp` 然后定位到这个页面.

这里的大部分内容由 Matt Dowell <matt.dowell@notiva.com> 提供

06-15-2005 Documentation

This page last changed on Mar 30, 2006 by scud.

Attendees

- Erik Beeson
- Simon Stewart
- Jason Carriera
- Patrick Lightbody
- Jay Bose
- Vitor Souza
- Gr gory Joseph

Outcome

- Team agrees that the three main guidelines should be:
  - Documentation and code must be kept in sync
  - Major sections should be focussed on different types of users: tutorials ("getting started"), reference, cookbook, general docs
  - WebWork documentation should be core focus, with XWork documentation being included in the WebWork docs where possible.
- Implementation strategies for these guidelines are:
  - Code/docs sync: utilize the Confluence snippet macro. We may need Atlassian to help get this macro in to shape.
  - Sections: each person in the meeting will write their own TOC and present it next week - see 06\-15\-2005 TOC Homework
  - Standaline WebWork docs: we will utilize the {include} macro to include whole XWork pages when possible, but we will avoid linking to the XWork docs.

Transcript

```
   Patrick_        hey
        shs96c  G;day
        greg--  hi
        Patrick_        jason should be on soon i imagine -- just saw him get on AIM
        shs96c  Fair enough.
        Patrick_        who do we have here? Rainer, Cameron, and who is Greg?
        shs96c  I'm Simon
        greg--  i'm sorry guys I was just passing by; badly need some sleep, it's 3 am here..
just saw the thread and thought i'd drop by
        greg--  I'm Gr#gory Joseph
        -->| jcarreira (~chatzilla@cpe-66-66-7-68.rochester.res.rr.com) has joined #webwork
        greg--  being noisy on the lists from time to time
        shs96c  Not a bad thing :)
        greg--  sent a couple of patches, and overall happy user of ww ;)
        Patrick_        ok, cool
        Patrick_        Greg: get some rest -- we'll post the transcript
        greg--  thanks ;)
        Patrick_        unfortunately, I wanted to have a rough TOC for the new docs that we
could all start with, but I didn't get around to it
        greg--  am waiting for a maven build to finish... it's moving at its own pace :/
        jcarreira       serves you right for using maven :-)
        Patrick_        Jason: could you maybe take lead on this meeting? I want to finish the
build refactoring I've been doing all day. Also, are we waiting for Jay, or do you want to get
started?
        shs96c  I'm happy to wait for Jay
        jcarreira       let's wait a couple more minutes
        Patrick_        ok. while we're waiting, i have some good news: the XWork build is the
first official project to be based on the OpenSymphony Common Build system
        Patrick_        WebWork is being updated now. Part of that update involves making
sub-projects in WW. The first such subproject will be the example web app
        shs96c  Patrick: What's the "common build system"?
        shs96c  Is it in CVS yet?
        Patrick_        http://ivyrep.opensymphony.com/opensymphony/ is where builds that use
the OpenSymphony Common Build drop their artifacts. This allows you to keep up to date if you
use Ivy
        Patrick_        yes, it is
        Patrick_        let me get you a URL
        Patrick_
https://xwork.dev.java.net/source/browse/xwork/build.xml?rev=1.30&view=auto&content-type=text/vnd.viewcvs-ma
        Patrick_        and:
        Patrick_
https://opensymphony.dev.java.net/source/browse/opensymphony/common/osbuild.xml
        Patrick_        a couple changes came from this:
        Patrick_        1) you must have ../opensymphony/common available
        shs96c  I was just about to ask about that
        Patrick_        (or you can redefine the location via ${common.build}
        shs96c  Can it be a jar?
        shs96c  Or is it the full source tree?
        Patrick_        it is just an ant build file
        Patrick_        it can't be a jar
        Patrick_        2) in XWork, I'm not currently building the editor. I can set this up
as a sub project, but I'm not sure if it is even worth it. Is the editor still maintained, or
should we focus our energy onConductor/EclipseWork/ etc?
        -->| vitorsouz (~vitorsouz@201.29.8.92) has joined #webwork
        shs96c  I've never used the editor, so not too stressed either way
        vitorsouz       Hi, there. Sorry I'm late.
        Patrick_        vitor: no worries, we haven't started yet.
        shs96c  NP
        Patrick_        Jason: maybe we should start now w/o Jay?
        vitorsouz       I wasn't sure if Eastern was DST or not. Is it 9:15 or 10:15 there now?
        Patrick_        it is 9:15 EST
        shs96c  11:15AM in Sydney. A very civil time to arrange a meeting on IRC for :)
        vitorsouz       10:15PM in Brazil. Not that different.
        Patrick_        ok, well let's start
        Patrick_        i'd like to start off with a couple thoughts, and then i'll open it up
        jcarreira       oops, back
        Patrick_        Recently, Matt Raible pointed me to the WebWork 1.0 Release
Announcement on TheServerSide
        Patrick_        more than a couple comments mentioned how good the documentation was
        jcarreira       LOL
        shs96c  :)
        Patrick_        whether it is just PR or reality, the fact is: WebWork suffers from an
image of bad documentation
        shs96c  Agreed
        Patrick_        the goal should be to fix that for WebWork 2.2
        jcarreira       yep
        Patrick_        in order to do that, I think we should establish some general
guidelines, and then once we agree on those, put together a TOC and divvy out the work
        jcarreira       ok
        vitorsouz       Ok.
        Patrick_        guidelines might be, for example, that the WebWork docs are "self
contained", meaning they don't refer to the XWork docs. Or, they might mean that all example
code must be verified to work and reproduced in the example app. Whatever.
        shs96c  Sounds reasonable

        Patrick_        I think for this meeting, we should establish those guidelines and put
together a rough TOC
        Patrick_        Jason, anything to add? Otherwise, I'm ready to open it up to
discussion of guidelines
        greg--  (you might want to use the snippet macro for confluence, to make sure sample
code in docs matches buildable reality - i.e. extract that code in the doc right out of cvs)
        jcarreira       well, I'd say another guideline should be that tutorials on the example
app should be documented in the doc
        jcarreira       greg, any idea what happens with that when you export that page?
        Patrick_        ok, sounds like greg and jason are sort of pointing to the same goal:
keep the code and docs in sync
        jcarreira       will it pull the latest code and include it?
        greg--  jcarreira: i guess like any macro, it just exports whatever is rendered, i.e.
the code as it is on cvs at the moment you export/publish
        jcarreira       I'd also say I think we need to think about who the audience is and
have a couple (at least) sets of docs... like a 5 minute quick start vs. tutorials / example
app vs. reference
        greg--  never verified that myself though
        shs96c  Jason: the most important docs are those aimed at beginners
        shs96c  Those are the people who need the most support
        Patrick_        OK, i'm going to keep a list of ideas being posted here. If I don't
summarize it properly, let me know.
        Patrick_        So far:
        Patrick_        - Keep docs and code in sync
        jcarreira       shs96c: well, that's one important set, but a complete and up-to-date
reference is important for current users (or even me)
        Patrick_        - Partition the docs properly: getting started, advanced users, etc
        shs96c  Agreed, but if there were holes in the docos for advanced users it would be
understandable
        vitorsouz       I think that WW docs should read more like a book, starting with the
easy stuff and incrementing gradually, showing examples and explaining. This is the best
approach for begginners, I guess.
        vitorsouz       That's what I tried with the tutorial, a long time ago, and then didn't
have the time to improve it (*sigh*).
        jcarreira       vitor, that's important, but I spend a lot of time on the boards
pointing people to the reference on the tags
        jcarreira       Jay IM'd me and he's having a hard time getting into EFNET
        vitorsouz       Ok. Reference is also desirable, as well as a cookbook: advanced tasks.
        vitorsouz       Jason: I got into irc.ca.efnet.info with no problem.
        shs96c  vitorsouz: sounds like how I'd expect the docs to be structured
        Patrick_        ok, so i'm hearing needs for: reference, tutorials, getting started,
and a cookbook (which is just tips and tricks, right?)
        vitorsouz       Right. Gettint started and tutorial could be the same thing.
        Patrick_        what about keeping the docs self-contained by not linking out to XWork?
        shs96c  A reasonable level of self-containment would be good
        vitorsouz       Self-contained: I think it's a good idea. Referencing to XWork gives an
idea of incompleteness.
        jcarreira       Well, I agree in general... but maybe we could use the Confluence stuff
to pull in parts from XWork?
        shs96c  Allows people commuting and reading offline to get the most out of it
        greg--  Patrick_ that's a good point, but on the other you don't want to duplicate
stuff.. there's currently some duplication and confusion, for instance with the i18n or
validation docs that are both in ww and xw
        Patrick_        Jason: I agree, if we can find ways to re-use, that is great. But I
think the final result should be self-contained
        shs96c  Perhaps a quick glossing over of some of the important XW topics would be
beneficial with links to the official XW docs
        Patrick_        so, related to that question: do you guys think WebWork docs should get
the majority of the focus, with XWork as an afterthought?
        jcarreira       shs, that doesn't help when we export the docs and include them in the
distro
        vitorsouz       Patrick: yes.
        shs96c  jcarreira: agreed, but it's enough for someone to keep reading
        vitorsouz       XWork is more targeted to developers, I guess. They can read the source
code. ;)
        shs96c  Not so sure about that.
        vitorsouz       (just kidding)
        shs96c  :P
        Patrick_        ok, any other guidelines? we have three right now:
        Patrick_        - Keep docs and code in sync
        greg--  it would maybe hide xwork even more. as of now, it's not obvious what you do
with xw WITHOUT ww, and maybe you guys also want to stress that?
        Patrick_        - Break up in to main sections: tutorials (getting started, etc),
reference, cookbook
        Patrick_        - WebWork docs core focus, XWork docs not important and kept separate
        jcarreira       well, I'm not sure about that last one
        shs96c  I'm with Jason on this one
        Patrick_        ok, what do you suggest?
        shs96c  I'd like the XW docs to be "the source of truth" for XW things
        jcarreira       I think we should look at the ability to keep the XWork docs up to date
and use Confluence to include them in WebWork
        vitorsouz       Here's another option: both XWork and WebWork (and maybe future
projects derived from XWork) have the same docs.
        Patrick_        I agree, but not at the expense of the WebWork documentation experience
        shs96c  I'd be happy to see some high level discussion of how the XW elements affect
WW2 in the WW2 docs
        -->| nightfal (~nightfal@c-67-180-134-149.hsd1.ca.comcast.net) has joined #webwork
        -->| jaybose (~chatzilla@CPE-65-27-76-47.mn.res.rr.com) has joined #webwork
        jaybose sorry i'm late
        vitorsouz       Welcome Jay and nightfal.
        Patrick_        Jay, Erik, welcome
        shs96c  We're talking about documentation
        jcarreira       ok, guidelines so far:
        Patrick_        read here to catch up:
http://wiki.opensymphony.com/display/WW/Temp+chat
```

```
        Patrick_        let's nail down these three guidelines and then move on the TOC
        jcarreira       - Keep docs and code in sync
        jcarreira       to do that let's look at the stuff that pulls from CVS
        jcarreira       - Break up in to main sections: tutorials (getting started, etc), 
reference, cookbook
        jcarreira       I think what we have now is mostly reference, but not so well 
structured
        nightfal        I agree
        greg--  jcarreira : 
http://confluence.atlassian.com/display/CONFEXT/Snippet+Macro+Library
        jcarreira       Now the one we're not so much in agreement on: - WebWork docs core 
focus, XWork docs not important and kept separate
        shs96c  nod
        jcarreira       greg, cool.. I'll take a look...
        shs96c  That's the one I'm feeling a little uncomfortable about
        Patrick_        well, here's my two concerns:
        jcarreira       I'd like to keep XWork up to date, and I think Confluence can bring it 
all together
        nightfal        obviously it gets tricky because *most* of the use that xwork sees is 
from webwork, so separating them just confuses new webwork users
        Patrick_        1) I don't want our focus on XWork docs, since 99% of users will never 
download XWork
        shs96c  Except as part of WebWork.
        shs96c  :)
        Patrick_        2) I want the WebWork docs to always be correctly "versioned" -- 
meaning that when you download WebWork 2.2 and the docs point out to XWork, they can't point to 
the wiki
        Patrick_        because the wiki would be "latest", not necessarily the docs that we 
meant to link to at the time 2.2 was released
        jcarreira       right, agreed
        shs96c  We could always export the XWork and WebWork spaces together for the WebWork 
docs
        jcarreira       greg, do you know the macros to pull in pieces of other pages?
        jaybose There are things that would better be explained under the XWork section
        vitorsouz       What about the idea of both sharing the same docs? I got no comments on 
the idea (maybe because it's very very bad... :)).
        Patrick_        I don't want users to get fragemented in to thinking about XWork and 
WebWork. I want them to download WebWork and just use it. That means I don't want them to have 
to read an XWork Reference and a WebWork Reference
        Patrick_        vitor: i don't like that for the reasons above
        shs96c  Then we're heading towards Vitor's idea....
        nightfal        I agree that separate is poor
        nightfal        is exporting both sets of docs for the ww dist an option?
        Patrick_        I'm open to finding a way to include parts of the XWork docs in the 
WebWork docs via Confluence. But the end result would be *standalone* WebWork docs when 
exported
        jcarreira       yeah, I think what makes sense is to pull in the parts of the xwork 
docs that are needed and add to them in the corresponding webwork docs page
        Patrick_        is everyone else cool with that?
        nightfal        I'm stepping away for a minute, I just wanted to lurk
        nightfal        lurks
        shs96c  Standalone is fine by me, because I like to read offline
        Patrick_        ok, speak now or forever hold your peace :)
        Patrick_        going once... twice...
        jaybose so the plan is to just reference parts of the Xwork docs?
        vitorsouz       Wait...
        Patrick_        waiting :)
        jaybose but to keep sep?
        greg--  jcarreira : you mean {include} i believe ?
        vitorsouz       What about not mentioning XWork in the Getting Started (Tutorial) and 
referencing it in the reference and cookbook.
        vitorsouz       That way begginners wouldn't feel confused.
        jaybose i like that idea
        jcarreira       yes, I think that's a good idea
        shs96c  Beginners often think of WW2 and XW as one and the same thing...
        Patrick_        vitor: i'm fine with referencing it as long as the _content_ that is 
exported appears as a standalone document. The way we would do that is by using {include} (I 
think)
        vitorsouz       Ok. I'm fine with include.
        Patrick_        So, just so we're all absolutely clear, an example of this in action 
might be:
        vitorsouz       I'm more concerned with the end result for the reader, because I don't 
know Confluence and its features that well.
        Patrick_        XWork/Documentation/Interceptors/Overview might talk about interceptors 
in general
        greg--  i zlso it's better to mention it right away, than letting users discover the 
existence of xw once they think they're up to speed with ww
```

```
        Patrick_        WebWork/Documentation/Interceptors/Overview might link to 
WebWork/Documentation/Interceptors/XWorkOverview
        Patrick_        and then WebWork/Documentation/Interceptors/XWorkOverview would 
_include_ XWork/Documentation/Interceptors/Overview
        greg--  mind that {include} includes a complete page, not portions of it
        vitorsouz       Greg: we could mention it in the begginning of the tutorial, just to 
let the reader know, but saying they shouldn't worry about it for now.
        Patrick_        ok. say "wait" in the next 5 seconds or we're moving on
        greg--  vitorsouz true :)
        jcarreira       link?
        vitorsouz       I'm cool with that last resolution.
        jcarreira       you mean it links now and now we want to make it include?
        greg--  jcarreira 
http://docs.codehaus.org/renderer/notationhelp.action?section=confluence ?
        Patrick_        ok, great. so now the next step is TOC -- or maybe more specifically, 
how do we implement these gaols.
        Patrick_        let's start with the first one:
        Patrick_        KEEP THE CODE AND DOCS IN SYNC
        Patrick_        it appears that Simon and Jason are on to something that might help
        Patrick_        what I've been doing is making the example app an actual tutorial, with 
the tutorials _in_ the app
        Patrick_        it hasn't turned out too hot so far though :(
        jcarreira       yes, we need to use the snippet macro... what do we have to do to 
enable the snippet macro?
        shs96c  Sounds hard to do nicely
        jcarreira       well, it just needs the docs written, I think
        vitorsouz       Patrick: maybe more thought is to be given to which examples are 
interesting. But that's not the point right now, I guess. The point is the means of syncing, 
right?
        jcarreira       the code can't stand alone for someone trying to learn
        Patrick_        well, wait -- do we need to do te snippet macro? What about just saying 
that parts of the docs (say, "tutorials") are in the example app and the reference and cookbook 
are static?
        Patrick_        or would that fragment things too much b/c then you wouldn't be 
authoring all the docs in confluence?
        jcarreira       Hmm... so just do the tutorials completely in the example app?
        Patrick_        maybe, i'm just tossing out ideas at this point. i don't feel strongly 
either way
        vitorsouz       So the tutorial page in confluence would be just a single one, pointing 
to the example app in the distribution?
        Patrick_        possibly. would that work?
        Patrick_        how does {snippet} work?
        greg--  jcarreira : enabling the snippet macro = dropping the jar in 
confluence/WEB-INF/lib and praying it works with your confluence version.
        shs96c  I'm not keen on that idea
        jaybose yes, all tutorials via the example app
        shs96c  Maybe I've grabbed the wrong end of the stick here
        vitorsouz       I prefer the automatically syncing idea, but not sure it's feasible or 
easy.
        shs96c  If we're talking about including snippets from the example app in the 
tutorials, that's cool
        jcarreira       the problem there is that to update the tutorial you'd have to be a 
webwork developer with CVS write access
        Patrick_        jason: good point
        greg--  Patrick_ http://confluence.atlassian.com/display/CONFEXT/Snippet+Macro+Library
: it grabs contents of your cvs repo through a cvsview, between given markers (like START 
SNIPPET FOO, END SNIPPET FOO)
        Patrick_        though, we could provide "Documentation" access to webwork/tutorial in 
CVS
        Patrick_        java.net does support that
        jaybose is that really necessary?
        jcarreira       but that only gives access to the /web directory, doesn't it?
        Patrick_        hmm, the snippet macro looks like it can pull parts of the content. I 
like it.
        greg--  Patrick_ it can indeed
        jaybose is copy and paste that error-prone?
        vitorsouz       About the snippet thing: it guarantees automatically updating existing 
code, but if I write a new code in the example app I have to write a new page for it and use 
the snippet-tag, is that correct?
        Patrick_        jason: well, after tonight, webwork/tutorial will have all the source, 
libs, etc. it'll be it's own project
        Patrick_        vitor: yes
        jcarreira       jay, copy and paste doesn't keep things up to date
        Patrick_        i actually really like the snippet idea
        Patrick_        what are potential "gotchas" with it?
        greg--  xml
        greg--  maybe there's been a new version but last time i tried
        greg--  few month agos
        greg--  an xml snippet wouldn't render corretly
        vitorsouz       I'm thinking that bugs in the snippet tag are the only concern. If it
works well, no worries.
        greg--  i mean, it was readable, but the xml colouring was messed up
        Patrick_        we could get Atlassian to fix that I bet
        Patrick_        or even make a copy for ourselves
        Patrick_        that works :)
        greg--  hmm it's not maintained by them
        Patrick_        any other showstoppers? so far i've heard:
        Patrick_        - new tutorials in CVS will need to have the wiki get updated
        Patrick_        - snippet macro could be buggy
        greg--  (mind that maybe with conf1.4 it'd work, they've rewritten the renderer)
        Patrick_        we're running on 1.4.1
        greg--  i can't really tell, we're still on 1.3.x at work
        Patrick_        ok, anything else to say about? any other suggestions? say "wait" in
the next 5 seconds or I'll assume we're agreed to use the snippet tag
        Patrick_        ok, done. (trying to keep this meeting plodding right along)
        vitorsouz       Right.
        Patrick_        next up: sections
        Patrick_        so we've talked about tutorials, reference, and cookbook
        Patrick_        where would something like the general description of WW's architecture
go?
        jaybose what goes in the three
        Patrick_        three?
        jaybose we need guidelines for that
        jaybose three sections.
        Patrick_        i'm not following
        Patrick_        you mean it would be in the cookbook?
        jaybose ok, what would go in the cookbook, and not in the tutorials
        vitorsouz       I think the docs should start with general information and
architecture. Then explain the three sections above: Tutorials for new users, Reference for
quick ref and Cookbook for advanced users looking for advanced ideas.
        Patrick_        well, i see tutorials as things like:
        jcarreira       the tutorials are really 2 sections: getting started and tutorials
        shs96c  Things like injecting services into validators using Spring....\
        Patrick_        - Getting Started, Using Validation, etc
        Patrick_        the cookbook would b emore like:
        Patrick_        - How to override the default XHTML templates
        Patrick_        - Writing your own ServletDispatcher
        Patrick_        etc
        vitorsouz       I see the tutorials as something that starts with installation and
"Hello, World" and slowly increment things until we have a complete but simple example app.
        jaybose getting started should be part of the ref manula
        shs96c  vitorsouz: I like that idea
        Patrick_        see, i think we're missing something: general documentation
        jcarreira       yeah
        Patrick_        i think we need a "setting up webwork" in the general docs
        Patrick_        and then in the tutorials, a "getting started"
        Patrick_        there is a difference:
        jcarreira       ok docs:
        Patrick_        in "setting up webwork", you're told what configs are needed, files,
etc
        Patrick_        in "getting started", it is a step by step hand-holding guide
        vitorsouz       Gettint Started is in the tutorials, Setting up WebWork is in the
reference.
        jcarreira       let's start at the top level and then break each one down
        jcarreira       we need: Getting started (includes a hello world starter app and
tutorials)
        vitorsouz       Just to further explain my point of view, once the user finishes the
tutorial he/she could go to the Cookbook and check random advanced ideas, meaning that cookbook
"mini-tutorials" do not depend on one another, but depend on know what's in the tutorial.
        Patrick_        ok, let's look at a couple things:
        jcarreira       General Documentation: includes architecture guide, what is MVC,
reference
        Patrick_        WebWork 1.x docs: http://www.opensymphony.com/webwork_old/
        shs96c  Something like: http://wiki.rubyonrails.com/rails/show/UnderstandingRails
        shs96c  ?
        Patrick_        WebWork 2.x docs:
http://www.opensymphony.com/webwork/wikidocs/Documentation.html
        [ERROR] Connection to irc://efnet/ (irc://irc.prison.net/) reset.
        =-=     User mode for Patrick__ is now +i
        -->| YOU (Patrick__) have joined #webwork
        |<-- Patrick_ has left efnet (Read error 54: Connection reset by peer)
        Patrick__       sorry, got booted
        =-=     YOU are now known as Patrick_
        Patrick_        ok, looking at our existing TOC, i think we're not too far from where
```

```
we want to be
        Patrick_        I think the real problem is this:
        vitorsouz       "Understanding Rails" is nice.
        Patrick_        http://www.opensymphony.com/webwork/wikidocs/Interceptors.html
        Patrick_        you click on Interceptors
        Patrick_        but now you can't get any useful info on the "prepare" interceptor, for
example
        Patrick_
http://www.opensymphony.com/webwork/wikidocs/ExecuteAndWaitInterceptor.html is what we need to
aim for for every reference page
        Patrick_        something more detailed
        Patrick_        notice that XW has more info on some of the items:
        Patrick_        http://wiki.opensymphony.com//display/XW/Interceptors
        Patrick_        ok, it's getting late for everyone, I think we should wrap this up
rather than letting this go on for another hour. can someone volunteer to write a new TOC that
at least shows examples of drilling down to the details needed in the reference?
        jcarreira       Yes, some of the WebWork pages for interceptors may ONLY have an
include of the XWork page
        vitorsouz       Sorry to return to the XWork/WebWork docs issue, in this case, we would
have a XWork Interceptors section including XWork docs and then a WebWork-specific Interceptors
section with WW stuff?
        greg--  i'll be the 1st to leave - darn .. 4pm :d
        jcarreira       ok, thanks for the help greg!
        greg--  ur welcome :)
        Patrick_        vitor: there would be a page, for example, in WebWork called
PrepareInterceptor
        greg--  bye everyone
        |<-- greg-- has left efnet ()
        Patrick_        and it may simply just include XWork's PrepareInterceptor page
        Patrick_        the key is that it includes it rather than linking to it
        |<-- jaybose has left efnet (Ping timeout: no data for 247 seconds)
        vitorsouz       Hmmmm... Ok.
        Patrick_        we have to be careful about links in the XWork docs though
        -->| jaybose_ (~chatzilla@CPE-65-27-76-47.mn.res.rr.com) has joined #webwork
        =-=     jaybose_ is now known as jaybose
        Patrick_        however, I think that we can sort of think of these things after the
initial docs are there
        Patrick_        Jay, i'll update the chat log
        vitorsouz       I'd volunteer to write the TOC, but I'm going out of town tomorrow,
only returning Sunday. If you guys don't mind waiting for some time next week...
        jcarreira       Jay, do you have time to work on the TOC?
        jaybose yes
        jcarreira       Ok, you can put it under the WebWork 2.2 page on the wiki... it doesn't
have to link to anything yet
        Patrick_        http://wiki.opensymphony.com/display/WW/Temp+chat
        vitorsouz       Seems like some people have different ideas in general for the doc
structure, how about we schedule the next meeting and everyone writes a TOC exemplifying his
own ideas and sends it to Patrick to put online for discussion?
        Patrick_        vitor: i like that idea. and those who don't write one don't get as
much of a say :)
        jcarreira       :-)
        vitorsouz       :)
        shs96c  :)
        jcarreira       ok, when are you back from your trip Vitor?
        vitorsouz       Late saturday night. I could work on this sunday.
        Patrick_        haha, sounds like we have a winner then. Let's schedule the next
meeting time and call it a day
        shs96c  Same time on Sunday?
        Patrick_        Sunday is too early
        shs96c  OK
        Patrick_        i'd opt for sometime after Tuesday next week.
        jcarreira       next wed?
        Patrick_        same time next week?
        nightfal        Same bat time, same bat channel?
        shs96c  Fine by me
        vitorsouz       Alright. Wednesday 9PM Eastern.
        jcarreira       yep, sounds like a plan
        nightfal        TOC == Table of Contents, yes?
        Patrick_        yes
        vitorsouz       Yes.
        Patrick_        great! good work guys. this was a perfect meeting too -- 60 minutes and
done :)
        Patrick_        so next week we'll nail down the TOC and divide up responsibilities
        jcarreira       yep
        vitorsouz       Okidoki.
        Patrick_        i'll post the final chat log and send out a note in the forums for
people interested
```

```
Patrick_        see ya. good meeting
jcarreira       ok, sounds good
|<-- jaybose has left efnet (Chatzilla 0.9.68.5 [Netscape 7.2/20040804])
vitorsouz       Good idea. Great work, everyone.
jcarreira       later
```

This page last changed on Mar 30, 2006 by scud.

Doc Team Suggestions

Documentation Style

Click here for the suggestions on documentation style.

Documentation:

One-paragraph description of what is WebWork. And then, an explanation of how the documentation is divided:

If you're new to WebWork, please read the **Overview** and proceed to the **Tutorial** to get started. Experienced users can refer to the **Cookbook** for advanced topics. Use the **Reference** on an as-needed basis for more specific details. For detailed information about WebWork project, read the section **Project Information**. Information about many projects related to WebWork can be found in **Related Projects**

If you have any questions, you can ask them at the user forum/mailing list. Please be sure to read the **FAQ** before asking any questions.

1. Overview
2. Project Information
3. FAQ
4. Tutorial
5. Cookbook
6. Reference
7. Related Projects

(details on each of the above follows)

Overview

1. What is WebWork
2. Comparison to Struts (Tapestry, JSF, etc.?) - current material and links to existing articles
3. Articles and press about WebWork
4. Projects using WebWork / Testimonials

Project Information

1. License
2. Deployment notes
3. WebWork versions
    - Current release
    - Previous releases
    - Migrating from WebWork 1.x;
4. Dependencies
5. WebWork Team

6. WebWork community
    - Mailing lists / Forum
    - Bug tracker
    - Wiki
    - How to contribute?

FAQ

1. Category 1
2. Category 2
3. Etc.

The questions included in the FAQ should be extracted from the User Forum/Mailing List. Someone could review recent messages and find out which questions are asked the most. After that, separate the questions into categories to form the subsections.

Tutorial

1. Downloading and installing WebWork
2. Setting up the test environment (to test tutorial source code)
3. Basic configuration and your first action (Hello WebWorld)
4. Understanding actions
5. Understanding results
6. Meet WebWork tag library (would also explain a little bit of OGNL)
7. Evaluating other view options: Velocity
8. Evaluating other view options: FreeMarker
9. Understanding interceptors
10. Performing validation
11. Performing dependency injection (IoC) through components
12. Going i18n (internationalization)
13. Retrieving data without a full request using XHR/Ajax

This should be in conformance to Patrick's example app. Vitor will talk to Patrick to get his opinions on the sequence of lessons suggested above and how to make the tutorial conformant to the example app.

Cookbook

- Tips and tricks on Application Servers (this was in "Overview")
- Stuff from the current Cookbook...

All the tips in the Cookbook should be revised. Some of them could belong to the tutorial instead (basic stuff). 3rd party integration tips could be separated from the rest. Also, it would be good if all of them also followed the same structure, kind of like a tutorial lesson, but on advanced topics. The differences between the Tutorial and the Cookbook would be:

	Tutorial	Cookbook
User level required:	Begginner	Intermediate
Availability of Source code:	In the documentation and in the `src` folder of the distribution (ready for deploy and test).	Only in the documentation, and may not be complete.
Should be read in sequence?	Yes.	No.

A question that arised in the TOC discussion was: what's the difference between the Cookbook and the FAQ?

Well, some of the items in the Cookbook are also FAQs (people ask about them a lot), so they would also be included in the FAQ, with links to the Cookbook. The FAQ should have quick answers or link to longer answers, such as the Cookbook. The Cookbook is tutorial-style, a collection of mini HOW-TOs.

Reference

1. Introduction - This will have parts of Overview in it, rather than forwarding them to Overview altogether.
2. Architecture
3. Configuration
4. Interceptors
5. Action Chaining
6. IOC / Dependency Injection
7. UI Components - JSP, Velocity, Freemarker, JavaScript validation and DWR support
8. Result Types
9. Type Conversion
10. Validation
11. OGNL / Object Graph Navigation Language
12. Internationalization
13. 3rd Party Integration - Sitemesh, Spring, Pico, Hibernate, JUnit, Quartz, etc.

An important thing about the reference is that it should be written in book-style (or hibernate-reference-style) so a PDF version would be generated and people could print it, pass it around the office or read it while working out or relaxing in bed. 🙂 Therefore, Confluence's PDF features play a big part in this. Some questions that came up during the discussions:

- The reference could link to other pages in some situations. Links in confluence do not reference URLs, but the page's names. Does Confluence convert them to the URL before writing the PDF? If not, should the Reference not link to any pages outside it?

- Can we select which pages are converted into PDF? If we want to convert only the Reference to PDF, can we do that? How does that work?

Related Projects

1. WebFlow (graphical chart tool)
2. EclipseWork (Eclipse Plugin)
3. IDEA Plugin
4. WebWork Optional
5. Etc. ?

Space Details

Key:	ww2cndoc
Name:	WebWork2文档中文化计划
Description:	翻译的WebWork2的中文文档
Creator (Creation Date):	scud (Jan 10, 2006)
Last Modifier (Mod. Date):	scud (Mar 28, 2006)

Available Pages

        - web.xml 2.1.x compatibility
    - webwork-default.xml
    - webwork.properties
    - xwork.xml
- Continuations
- Cookbook
    - Access to Webwork objects from JSP 2.0 EL
    - Accessing application, session, request objects
    - Application, Session, Request objects in jsp
    - Application, Session, Request objects in vm
    - Describing a bean in velocity
    - Exposing webwork objects to JSTL, with a JSTL and DisplayTag Example
    - GroovyResult
    - Handing IoC Components to Interceptors and Validators
    - How do I populate a form bean and get the value using the taglib
    - How to format dates and numbers
    - How to validate field formats, such as a phone number
    - Interceptor Order
    - Iterator tag examples
    - JFreeChartResult
    - redirect after post
    - RomeResult
    - Setting up Eclipse with Tomcat
    - Tabular inputs with XWorkList
    - Transparent web-app I18N
    - Using Checkboxes
        - Using Checkboxes - EditAction.java
        - Using Checkboxes - User.java
        - Using Checkboxes - Velocity and HTML
    - Using Maven to set up an Eclipse project for Webwork
    - Using WebWork and XWork with JSP 2.0 and JSTL 1.1
    - Using WebWork Components
    - Value Stack Internals
    - Webwork 2 HTML form buttons Howto
    - Webwork 2 skinning
    - Webwork file upload handling
- Dependencies
- Documentation
- EclipseWork
- Examples
    - Asynchronous processing with WebWork - XWork
    - Chat Application
    - SimpleLogin with Session
- Exception Handling
- FAQ
    - Can I access my action's Result
    - Can I add I18N outside the Action's context
    - Can I break up my large XWork.xml file into smaller pieces
    - Can I change templateDir on a per-page basis
    - Can I change templateSuffix on a per-page basis
    - Can I change theme on a per-page basis
    - Can I enable ww altSyntax on a per-page basis
    - Can I have my webwork action tag run another method apart from the default

    execute method
    - How can I display image that are contained as bytes in my action
    - How can I get the HttpServletRequest
    - How can I get the HttpServletResponse
    - How can I integrate WebWork IoC in to an object that is not an action
    - How can I put a String literal in a Javascript call, for instance in an onChange attribute
    - How can I see all request parameters passed into the action
    - How do I add I18N to a UI tag, like the textfield tag
    - How do I change the error message for invalid inputted fields
    - How do I decouple XWork LocalizedTextUtil global resource bundle loading from serlvets
    - How do I get access to the session
    - How do I get JEE J2EE security info
    - How do I get static parameters into my action
    - How do I get the latest version of WebWork
    - How do I handle files upload
    - How do I populate my action properties upon validation failure
    - How do I set a global resource bundle
    - How do I use messages from within the validator
    - How to build the portlet war for a specific portal server
    - How to escape special chars in resource bundles
    - How to reload xwork configuration
    - How to support UTF-8 URIEncoding with Tomcat
    - I'm trying to run the webwork example in the tutorial on Tomcat, and it can't instantiate the VelocityEngine
    - What are the default variables in the value stack
    - Which portal servers are supported
    - Why am I getting RuntimeException saying Compound Root cannot find a particular Object with a particular property
    - Why didn't my webwork action tag gets executed when I have validation errors
    - Why do I get error message saying Attribute 'short-circuit' must be declared for element type 'field-validator'
    - Why does FreeMarker complains that there's an error in my user-directive when I used JSP Tag
    - Why does webwork keeps calling my action's properties (eg. my getModel if my action implements ModelDriven) multiple times
    - Why isn't my Prepare interceptor being executed
    - Why won't the 'if' tag evaluate a one char string
- FreeMarker
- IDEA Plugin
- Interceptors
    - Alias Interceptor
    - Chaining Interceptor
    - Component Interceptor
    - Conversion Error Interceptor
    - Create Session Interceptor
    - Exception Interceptor
    - Execute and Wait Interceptor
        - HibernateAndSpringEnabledExecuteAndWaitInterceptor
    - File Upload Interceptor
    - I18n Interceptor

- Logger Interceptor
- Model Driven Interceptor
- Parameter Filter Interceptor
- Parameters Interceptor
- Prepare Interceptor
- Scope Interceptor
- Servlet Config Interceptor
- Static Parameters Interceptor
- Timer Interceptor
- Token Interceptor
- Token Session Interceptor
- Validation Interceptor
- Workflow Interceptor

- Internationalization
- Inversion of Control
    - Components
    - IoC Configuration
    - IoC Overview
    - Xwork's Component Architecture
- J2SE 5 Support
    - After Annotation
    - AnnotationWorkflowInterceptor
    - Before Annotation
    - BeforeResult Annotation
    - Conversion Annotation
    - ConversionErrorFieldValidator Annotation
    - CreateIfNull Annotation
    - CustomValidator Annotation
        - ValidationParameter annotation
    - DateRangeFieldValidator Annotation
    - DoubleRangeFieldValidator Annotation
    - Element Annotation
    - EmailValidator Annotation
    - ExpressionValidator Annotation
    - FieldExpressionValidator Annotation
    - IntRangeFieldValidator Annotation
    - Key Annotation
    - KeyProperty Annotation
    - RegexFieldValidator Annotation
    - RequiredFieldValidator Annotation
    - RequiredStringValidator Annotation
    - StringLengthFieldValidator Annotation
    - StringRegexValidator Annotation
    - TypeConversion Annotation
    - UrlValidator Annotation
    - Validation Annotation
    - Validations Annotation
    - VisitorFieldValidator Annotation
- JasperReports
- JSP
- JUnit
- Misc
    - AJAX Validation - points for discussion

    - JasperReports Result
    - PlainText Result
    - Redirect Action Result
    - Redirect Result
    - Stream Result
    - Velocity Result
    - XSL Result
- Struts Action 2.0 Information
- Style Guide
- Tags
    - Control Tags
        - append
        - else
        - elseIf
        - generator
        - if
        - iterator
        - merge
        - sort
        - subset
    - Data Tags
        - action
        - bean
        - debug
        - i18n
        - include
        - param
        - property
        - push
        - set
        - text
        - url
    - Form Tags
        - checkbox
        - checkboxlist
        - combobox
        - datepicker
        - doubleselect
        - file
        - form
        - head
        - hidden
        - label
        - optiontransferselect
        - password
        - radio
        - reset
        - richtexteditor
        - select
        - submit
        - textarea
        - textfield
        - token

            - updownselect
        - FreeMarker Tags
        - JSP Tags
        - Non Form Tags
            - a
            - actionerror
            - actionmessage
            - component
            - date
            - div
            - fielderror
            - panel
            - tabbedpane
            - tabbedPanel
            - table
            - tree
            - treenode
        - Tag Syntax
            - Alt Syntax
        - Themes and Templates
            - ajax theme
                - ajax a template
                - ajax div template
                - ajax event system
                - ajax head template
                - ajax submit template
                - ajax tabbedPanel template
            - css_xhtml theme
                - css_xhtml form template
                - css_xhtml head template
            - Extending Themes
            - Selecting Themes
            - simple theme
                - simple head template
            - Template Loading
            - xhtml theme
                - xhtml form template
                - xhtml head template
        - Velocity Tags
    - Testimonials
    - The Vault
    - Tutorial
        - Basic configuration and your first action - Hello WebWorld
        - Create a webapp project structure for your web application
        - Getting Started
        - Lesson 1 - Setting up webwork in a web application
        - Lesson 2 - An html form with no data
        - Lesson 3 - An html form with data
        - Lesson 4 - An html form with data, without getters or setters
        - Portlet Tutorial
        - Tiles Use
        - TutorialLesson05
            - TutorialLesson04-01

            - TutorialLesson04-01-01
        - TutorialLesson04-02
        - TutorialLesson04-03
    - TutorialLesson06
    - Understanding actions
    - Understanding interceptors
    - Understanding results
    - Understanding validation
  - Type Conversion
  - Validation
    - AJAX Validation
    - Basic Validation
    - Client Side Validation
        - AJAX Client Side Validation
        - Pure JavaScript Client Side Validation
    - Client Validation
    - conversion validator
    - date validator
    - email validator
    - expression validator
    - fieldexpression validator
    - int validator
    - regex validator
    - required validator
    - requiredstring validator
    - stringlength validator
    - url validator
    - Using Field Validators
    - Using Non Field Validators
    - Using Visitor Field Validator
    - visitor validator
  - Velocity
    - velocity.properties
  - WebWork Overview
  - XWork Configuration
    - Action configuration
    - Include configuration
    - Interceptor Configuration
    - Namespace Configuration
    - Package Configuration
    - Result Configuration
        - Default results
        - Global results
    - Views
- WebWork Wiki Chinese Version 1.0a