

Space Details

Key:	WW
Name:	WebWork
Description:	
Creator (Creation Date):	plightbo (Apr 18, 2004)
Last Modifier (Mod. Date):	matthew (Feb 08, 2005)

Available Pages

- WebWork 
 - 3rd Party Integration
 - Acegi Security
 - Hibernate
 - AdminApp
 - Non-IoC version of OpenSessionInViewInterceptor
 - JSTL
 - Pico
 - Quartz
 - SiteMesh
 - Spring
 - Other Spring Integration
 - Spring Session Components Workarounds
 - WebWorkTargetSource Shopping Cart Example
 - Spring Webflow
 - Tiles
 - About WebWork
 - App Servers
 - Glassfish
 - SunOne 7.0
 - WebLogic
 - WebLogic 6.1
 - WebSphere
 - Articles and press
 - Strutting the OpenSymphony way
 - Companies that provide WebWork support
 - Comparison to other web frameworks
 - Comparison to JSF
 - Comparison to Ruby on Rails
 - Comparison to Spring MVC
 - Comparison to Struts
 - Comparison to Tapestry
 - Press Releases

- 2.1 Press Release
 - 2.1.1 Press Release
 - 2.1.2 Press Release
 - 2.1.3 Press Release
 - 2.1.4 Press Release
 - 2.1.5 Press Release
 - 2.1.6 Press Release
 - 2.1.7 Press Release
- About
- Previous releases
 - Release Notes - 2.1
 - Release Notes - 2.1.1
 - Release Notes - 2.1.2
 - Release Notes - 2.1.3
 - Release Notes - 2.1.4
 - Release Notes - 2.1.5
 - Release Notes - 2.1.6
 - Upgrading from 1.4
 - JSP Expression Language Comparison with WebWork 1.x
 - Upgrading from 2.0
 - Upgrading from 2.1
 - Upgrading from 2.1.1
 - Upgrading from 2.1.2
 - Upgrading from 2.1.3
 - Upgrading from 2.1.4
 - Upgrading from 2.1.5
 - WebWork 2.1.7
 - WebWork 2.2
 - WebWork 2.2 Migration Notes
 - WebWork 2.2.1
 - WebWork 2.2.2
 - WebWork 2.2.3
 - WebWork 2.2.4
 - WebWork 2.2.5
- Projects Using WebWork
- Struts 2.0 Information
- Testimonials
- WebWork Overview
- About XWork
 - Upgrading XWork
 - Upgrading from 1.0
 - Upgrading from 1.0.1
 - Upgrading from 1.0.2
 - Upgrading from 1.0.3
 - Upgrading from 1.0.4

- Upgrading from 1.1
- XWork 1.0.1
- XWork 1.0.2
- XWork 1.0.3
- XWork 1.0.4
- XWork 1.0.5
- XWork 1.0.6
- XWork Press Releases
 - 1.0.1 Press Release
 - 1.0.2 Press Release
 - 1.0.3 Press Release
 - 1.0.4 Press Release
 - 1.0.5 Press Release
 - 1.1.2 Press Release
 - 1.2.0 Press Release
 - 1.2.2 Press Release
- Continuous Build of WebWork and XWork
- Developers
 - Building WebWork
 - Style Guide
- Documentation
- Help
 - Chat
 - Meetings Minutes
 - 06-01-2005 AJAX
 - 06-15-2005 Documentation
 - 06-15-2005 TOC Homework
 - 07-04-2005 Documentation
- 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 get the ServletContext
 - 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 allow WebWork action to be executed only using WebWork Action Tag but not from direct url access from browser
- How do I change the error message for invalid inputted fields
- How do I decouple XWork LocalizedTextUtil global resource bundle loading from servlets
- 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 date into datepicker component
- How do I set a global resource bundle
- How do I unit test my action's validation logic
- 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 want my form's submit button to be displayed differently eg. besides my text field
- I'm trying to run the webwork example in the tutorial on Tomcat, and it can't instantiate the VelocityEngine
- Internet Explorer showing a prompt saying 'This page contains both secure and nonsecure items' when using dojo
- Problem getting SiteMesh to work with web.xml errorPage
- Some container complains about OGNL expression while parsing my jsp page
- WebWork failed to grab parameters of sort link generated by DisplayTag
- What are the default variables in the value stack
- What should I do about the conflict with JSP 2.1 EL and WebWork OGNL EL
- 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 do some validators like stringlength ignore validation when input is empty, it worked before

- Why does FileUpload ignore the file size rule specified
- Why does FreeMarker complains that there's an error in my user-directive when I used JSP Tag
- Why does my FileUpload not work
- Why does my setter not get called
- Why does Url tag compound my namespace
- Why does WebWork fail to find some javascript
- Why does WebWork failed when trying to validate my xml configuration file against its dtd
- Why does webwork keeps calling my action's properties (eg. my getModel if my action implements ModelDriven) multiple times
- Why does WebWork only manage to retrieve attribute from HttpSession the second time around
- Why does WebWork's Rich Text Editor does not render
- Why does WW ignore my message when its enclosed in CDATA
- Why doesn't WebWork's If tag evaluate test="
- Why isn't my Prepare interceptor being executed
- Why won't the 'if' tag evaluate a one char string
- The Vault
- JSP 2.1 Expression Language And WebWork Tag
- Misc
 - AJAX Validation - points for discussion
 - CeWolf charts using Velocity templates
 - 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
 - File Upload Interceptor
 - 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
- Developing WW Ajax Widgets
- DHTML and XMLHttpRequest References
- HttpSession in Action
- JavaPolis Opensource Dinner 2006
- Learn WW by example
- Multiple Submit Buttons
- Performance Tuning
- Tabular inputs with HashMap
- Portlet Support
 - Portlet Configuration
- Reference
 - Action Chaining
 - ActionMapper
 - Architecture
 - Configuration Files
 - Reloading configuration
 - web.xml
 - web.xml 2.1.x compatibility
 - webwork-default.xml
 - webwork.properties
 - xwork.xml
 - Continuations
 - Dependencies
 - Exception Handling
 - FreeMarker
 - Interceptors
 - Alias Interceptor
 - Chaining Interceptor
 - Component Interceptor
 - Conversion Error Interceptor
 - Cookie Interceptor
 - Create Session Interceptor
 - Debugging Interceptor
 - Exception Interceptor

- Execute and Wait Interceptor
 - HibernateAndSpringEnabledExecuteAndWaitInterceptor
 - Flash Interceptor
 - I18n Interceptor
 - Logger Interceptor
 - Model Driven Interceptor
 - Parameter Filter Interceptor
 - Parameter Remover Interceptor
 - Parameters Interceptor
 - Prepare Interceptor
 - Scope Interceptor
 - Servlet Config Interceptor
 - Session Invalidation 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
- OGNL
 - OGNL Basics
- PreResultListener
- Result Types
 - Chain Result
 - Dispatcher Result
 - Flash Result
 - FreeMarker Result
 - WebWork Freemarker Support
 - HttpHeaders Result
 - JasperReports Result
 - PlainText Result
 - Redirect Action Result
 - Redirect Result
 - Stream Result
 - Velocity Result
 - XSL Result
- 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
 - fielderror
 - file
 - form
 - head
 - hidden
 - label
 - optgroup
 - optiontransferselect
 - password
 - radio
 - reset
 - richtexteditor
 - select
 - submit
 - textarea
 - textfield
 - token
 - updownselect
- FreeMarker Tags
- JSP Tags
- Non Form Tags
 - a
 - actionerror
 - actionmessage
 - component
 - date
 - div
 - 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
- Type Conversion
- Validation
 - AJAX Validation
 - Basic Validation
 - Client Side Validation
 - AJAX Client Side Validation
 - Workings Of WebWork Ajax Validation
 - Pure JavaScript Client Side Validation
 - Client Validation
 - collection validator
 - 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
- XWork Configuration
 - Action configuration
 - Include configuration
 - Interceptor Configuration
 - Namespace Configuration
 - Package Configuration
 - Result Configuration
 - Default results
 - Global results
 - Views
- Related Tools
 - Config Browser
 - Debugging inside WebWork - Debuggability of your Application
 - EclipseWork
 - IDEA Plugin
 - SiteGraph
 - ValueStack Explorer
- Tutorial
 - Basic configuration and your first action - Hello WebWorld
 - Create a webapp project structure for your web application
 - CRUD Demo I
 - Examples
 - Asynchronous processing with WebWork - XWork
 - Chat Application
 - SimpleLogin with Session
 - Getting Started
 - Installation
 - QuickStart
 - 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 IoC
- Understanding results
- Understanding tag libraries
- Understanding validation
- WebWork Continuous Integrations

WebWork

This page last changed on Apr 12, 2007 by [rainerh](#).

Welcome to the **WebWork** wiki. WebWork's official homepage is <http://www.opensymphony.com/webwork/>. There you can find documentation for the latest released version of WebWork. This wiki is used for additional information as well as documentation for the latest developing version.



This wiki is currently for WebWork **only**. For Struts 2.0 (which is based on WebWork), the wiki is located at <http://cwiki.apache.org/WW/home.html>

About WebWork

- [Features](#)
- [Release notes](#)
- [History, Team](#)
- [Contributing](#)
- [Bugs and issues \(direct\)](#)
- [Articles, Press releases, Testimonials, Projects using WebWork, Companies that provide WebWork support](#)
- [Downloads \(nightly builds, previous versions\)](#)
- [Comparison to other web frameworks](#)
- [License](#)

About XWork

- [Downloads \(nightly builds\)](#)
- [Press releases](#)
- [Upgrading XWork](#)

Getting Started

- [Installation](#)
- Requirements ([dependencies](#), [appservers](#))
- [Quickstart](#)
- [Tutorial, Another tutorial, Yet another tutorial 2](#)

Reference

- [WebWork API, XWork1 API](#)
- Basics
 - [Architecture](#)
 - [Configuration Files \(web.xml, xwork.xml, webwork.properties, webwork-default.xml, velocity.properties\)](#)
 - [XWork Configuration](#)
 - [Interceptors](#)

- [PreResultListener](#)
- [Result Types](#)
- [Exception Handling](#)
- [File uploading](#)
- Advanced
 - [Action Chaining](#)
 - [Inversion of Control \(IOC\)](#)
 - [Type Conversion](#)
 - [Validation](#)
 - [Internationalization](#)
 - [Continuations](#)
 - [ActionMapper](#)
- UI/Views
 - [Tags](#)
 - [JSP 2.1 Expression Language And WebWork Tag](#)
 - General ([Control Tags](#), [Data Tags](#))
 - HTML ([Form Tags](#), [Non Form Tags](#))
 - [OGNL \(syntax, altsyntax\)](#)
 - View technologies
 - [JSP \(specific tags\)](#)
 - [FreeMarker \(specific tags\)](#)
 - [Velocity \(specific tags\)](#)
 - [JasperReports](#)
 - [Themes and Templates](#)
 - [Components](#)
- [J2SE 5 Support \(Annotations\)](#)
- Portlets ([tutorial](#), [configuration](#))
- [Performance Tuning](#)

Users

- [Testing](#)
- [Code snippets](#) and [example applications](#)

Developers & Contributors

- [Style Guide](#)
- [WebWork source](#), [XWork1 source](#)
- [Ivy](#)
- [Building WebWork](#)

Integration, Tools, IDE

- IDE Plugins: [Eclipse](#), [IntelliJ](#)
- [Tools](#) ([SiteGraph](#), [Config Browser](#), [QuickStart](#), [ValueStack Explorer](#))
- [3rd party](#): [SiteMesh](#), [Spring](#), [Pico](#), [Hibernate](#), [JSTL](#), [JUnit](#), [Quartz](#), [Tiles](#), [Acegi Security](#), [Spring Webflow](#)

Help

- [FAQ & The Vault](#)
- [Mailing list](#)
- Forum
 - [WW Users \(direct\)](#)
 - [WW Development Team \(direct\)](#)
- [Chat \(Chat logs\)](#)

WebWork Documentation in Other Languages

- [Chinese Language](#)
Many thanks to Scud, who put in the initiatives and effort to make this possible.

Countinuous Integration

- [WebWork Continuous Integrations](#)
- [Continuous Build of WebWork and XWork](#)

3rd Party Integration

This page last changed on Jan 03, 2007 by [phil](#).

- [Acegi Security](#)
- [Hibernate](#)
- [JSTL](#)
- [Pico](#)
- [Quartz](#)
- [SiteMesh](#)
- [Spring](#)
- [Spring Webflow](#)
- [Tiles](#)

Acegi Security

This page last changed on Jan 03, 2007 by [phil](#).

The following details a possible integration of Acegi Security with WebWork:-

Step 1 - Declaring Authz Interface

```
import org.acegisecurity.taglibs.velocity.Authz;
public interface AuthzAware {
    void setAuthz(Authz authz);
}
```

Step 2 - Implementing Authz Interceptor

```
import org.acegisecurity.taglibs.velocity.Authz;
import org.acegisecurity.taglibs.velocity.AuthzImpl;
import package.api.AuthzAware;

import com.opensymphony.xwork.ActionInvocation;
import com.opensymphony.xwork.interceptor.Interceptor;

public class AuthzInterceptor implements Interceptor {
    public void destroy()
    {}

    public void init()
    {}

    public String intercept(ActionInvocation invocation)
        throws Exception
    {
        if (invocation.getAction() instanceof AuthzAware) {
            Authz authz = new AuthzImpl();

            AuthzAware authzAware = (AuthzAware)invocation.getAction();
            authzAware.setAuthz(authz);
        }

        return invocation.invoke();
    }
}
```

Step 3 - Making AuthzAware action

```
import org.acegisecurity.taglibs.velocity.Authz;
import package.api.AuthzAware;
import com.opensymphony.xwork.ActionSupport;

public class DashboardAction extends ActionSupport implements AuthzAware
{
    private Authz authz;

    public Authz getAuthz(){
        return authz;
    }

    public void setAuthz(Authz authz)
    {
        this.authz = authz;
    }
}
```

```
    }

    public String execute() throws Exception
    {
        return SUCCESS;
    }
}
```

Step 4 - Declaring interceptor

```
<interceptor name="authz" class="package.interceptor.AuthzInterceptor"/>
```

Step 5 - Declaring action

```
<action name="dashboard" class="package.action.DashboardAction">
    <interceptor-ref name="authz" />
    <result type="velocity" name="success">dashboard.vm</result>
</action>
```

Step 6 - Implementing dashboard.vm

```
Actualy you are logged as $authz.principal
```

Contributed by Luca Marrocco.

Hibernate

This page last changed on Mar 21, 2006 by [phil](#).

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](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 Jan 04, 2006 by [phil](#).

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

New syntax (since 2.2):

```
..  
<ww:form name="createForm" 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 execute, 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 structured, using as many advantages from the various frameworks as possible.

Non-IoC version of OpenSessionInViewInterceptor

This page last changed on Mar 21, 2006 by [phil](#).

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 Oct 30, 2005 by [plightbo](#).

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 Dec 30, 2005 by [plightbo](#).

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 Jun 14, 2005 by [plightbo](#).

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 Mar 14, 2007 by [tm_jee](#).

Overview

[SiteMesh](#) is a web-page layout and decoration framework and web- application integration framework to aid in creating large sites consisting of many pages for which a consistent look/feel, navigation and layout scheme is required.

Integrating WebWork with SiteMesh is amazingly simple: you don't have to do anything in fact. WebWork stores all its value stack information in the request attributes, meaning that if you wish to display data that is in the stack (or even the ActionContext) you can do so by using the normal tag libraries that come with WebWork. That's it!

ActionContextCleanUp

In WebWork's [Architecture](#), the standard filter-chain optionally starts with the **ActionContextCleanUp** filter, followed by other desired filters. Lastly, the **FilterDispatcher** handles the request, usually passing it on to the ActionMapper. The primary purpose of the **ActionContextCleanUp** is for SiteMesh integration. This tells the FilterDispatcher when exactly, to clean-up the request. Otherwise, the ActionContext may be removed before the decorator attempts to access it.



Warning

If ActionContext access is required within the decorators, the **ActionContextCleanUp** filter *must* be placed at the beginning of the filter-chain.

For more information, see the javadocs of the ActionContextCleanUp filter:

Special filter designed to work with the FilterDispatcher and allow for easier integration with SiteMesh. Normally, ordering your filters to have SiteMesh go first, and then FilterDispatcher go second is perfectly fine. However, sometimes you may wish to access WebWork-features, including the value stack, from within your SiteMesh decorators. Because FilterDispatcher cleans up the ActionContext, your decorator won't have access to the date you want. By adding this filter, the FilterDispatcher will know to not clean up and instead defer cleanup to this filter. The ordering of the filters should then be:

- this filter
- SiteMesh filter
- FilterDispatcher

Velocity and FreeMarker Decorators

WebWork provides extension of the SiteMesh PageFilter that assist with integration with [Velocity](#) and [FreeMarker](#). We strongly recommend using these filters, instead of the support provided by SiteMesh, because they also will provide the standard variables and [Tags](#) that you are used to when created views in your favorite template language.

Velocity

If you are using Velocity for your SiteMesh decorators, we recommend using the VelocityPageFilter. This is an extension of the SiteMesh PageFilter, which should be placed in the web.xml in between the **ActionContextCleanUp** and the **FilterDispatcher**. Now the Velocity decorators will have access to WebWork variables such as \${stack} and \${request}.

An error occurred:

<http://svn.opensymphony.com/svn/webwork/trunk/webwork/webapps/showcase/src/webapp/WEB-INF/web.xml?content-type=text/plain>
The system administrator has been notified.

FreeMarker

If you are using FreeMarker for your SiteMesh decorators, we recommend using the FreeMarkerPageFilter. This is an extension of the SiteMesh PageFilter, which should be placed in the web.xml in between the **ActionContextCleanUp** and the **FilterDispatcher**. Now the FreeMarker decorators will have access to WebWork variables such as \${stack} and \${request}.

An error occurred:

<http://svn.opensymphony.com/svn/webwork/trunk/webwork/webapps/showcase/src/webapp/WEB-INF/web.xml?content-type=text/plain>
The system administrator has been notified.

The following variables are available to the decorating freemarker page :-

- \${title} - content of <title> tag in the decorated page
- \${head} - content of <head> tag in the decorated page
- \${body} - content of t<body> tag in the decorated page
- \${page.properties} - content of the page properties

With the following decorated page :-

```
<html>
  <meta name="author" content="tm_jee" />
  <head>
    <title>My Title</title>
    <link rel="stylesheet" type="text/css" href="mycss.css" />
    <style type="text/javascript" language="javascript" src="myjavascript.js"></script>
  </head>
  <body>
    <h1>Sample</h1>
  </body>
</html>
```

Properties	Content
\${title}	My Title
\${head}	<link rel="stylesheet" type="text/css" href="mycss.css" /> <style type="text/javascript" language="javascript" src="myjavascript.js"></script>
\${body}	<h1>Sample</h1>
\${page.properties.meta.author}	tm_jee

Applying Freemarker decorator in tag form

Method 1: Using WebWork's Freemarker applydecorator Transform

This is the WebWork component that implements Freemaker's ApplyDecorator Transform. To use this Freemaker Transform, it needs to be enabled in webwork.properties (which is enabled by default)

```
webwork.freemarker.sitemesh.applyDecoratorTransform = true
```

An example of usage would be as follows:-

In Sitemesh's decorators.xml

```
<decorators defaultdir="/WEB-INF/decorators">
    ...
    <decorator name="panel" page="/panelDecorator.ftl" />
</decorators>
```

Decorator (panelDecorator.ftl)

```
<table border="1">
    <tr>
        <td>${title}</td>
    </tr>
    <tr>
        <td>${body}</td>
    </tr>
</table>
```

Freemaker page that uses decorator

```
<html>
<head>
    <title>some title</title>
</head>
<body>
    <h1>some body title</h1>
    <@sitemesh.applydecorator name="panel" page="/pages/pageToBeDecorated.ftl" />
</body>
</html>
```

An example of pageToBeDecorated.ftl

```
<html>
<head>
    <title>Panel Title</title>
</head>
<body>
    Panel Content
</body>
</html>
```

The nett outcome would be:-

```
<html>
    <title>some title</title>
    <body>
        <h1>some body title</h1>
        <table border="1">
            <tr>
                <td>Panel Title</td>
            </tr>
            <tr>
                <td>Panel Content</td>
            </tr>
        </table>
    </body>
</html>
```

```
</html>
```

The following are method hooks available to ApplyDecoratorBean and its subclass

- `getFreemarkerTemplate(String templatePath)` - create a Freemarker Template based on the template path given
- `parsePageFromContent(String content)` - returns a Sitemesh Page object based on the content as the to-be-decorated-page
- `parsePageFromAbsoluteUrl(String absoluteUrl)` - returns a Sitemesh Page object using the absoluteUrl to get the content of the to-be-decorated-page
- `parsePageFromRelativeUrlPath(String relativeUrl)` - returns a Sitemesh Page object using the relativeUrl to get the content of the to-be-decorated-page.
- `getSitemeshFactory()` - returns a Sitemesh Factory object
- `getPageParser(String contentType)` - returns a Sitemesh PageParser object
- `getDecorator(HttpServletRequest request, String decoratorName)` - returns a Sitemesh Decorator object with the decoratorName supplied.
- `deduceLocale(ActionInvocation invocation, Configuration freemarkerConfiguration)` - deduce the Locale from invocation else use the Locale supplied by Freemarker
- `createModel()` - create a Freemarker model for the template

Method 2: Using Sitemesh's ApplyDecorator Tag

Add Sitemesh's FreemarkerDecoratorServlet to web.xml

```
<servlet>
    <servlet-name>freemarkerDecoratorServlet</servlet-name>
    <servlet-class>com.opensymphony.module.sitemesh.freemarker.FreemarkerDecoratorServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>freemarkerDecoratorServlet</servlet-name>
        <url-pattern>*.dec</url-pattern>
    </servlet-mapping>
```

Add in Sitemesh decodator in decorator.xml eg.

```
<decorators>
    ...
    <decorator name="panel" page="panelDecorator.dec" />
</decorators>
```

Use Sitemesh's ApplyDecorator tag in page eg.

```
<#assign page=JspTaglibs['/WEB-INF/sitemesh-page.tld'] />
...
<@page.applyDecorator name="panel" page="/pageToBeDecorated.ftl" />
...
```

An example of pageToBeDecorated.ftl

```
<html>
    <head>
        <title>Some Title</title>
```

```
</head>
<body>
    Some Body
</body>
</html>
```

An example of panel.dec

```
<table border="1">
    <tr>
        <td>${title}</td>
    </tr>
    <tr>
        <td>${body}</td>
    </tr>
</table>
```

Nett effect after applying decorator

```
<html>
    ...
    <table border="1">
        <tr>
            <td>Some Title</td>
        </tr>
        <tr>
            <td>Some Body</td>
        </tr>
    </table>
    ...
</html>
```

Spring

This page last changed on Feb 12, 2007 by [tim_wonil](#).

Spring is, among other things, an Inversion of Control framework. As of WebWork 2.2, it is the recommended IoC container. You can find out more about Spring at <http://www.springframework.org>.



This section covers the only *supported* Spring integration technique. However, there are many other ways to tie in to Spring with WebWork. Please see [Other Spring Integration](#) for more info. Note that *none* of the other methods are currently supported and could change at any time!

Enabling Spring Integration

Turning on Spring support in WebWork is simply a matter of installing the latest Spring jars in to your classpath and then adding the following entry to [webwork.properties](#):

```
webwork.objectFactory = spring
```

If you want to change from the default autowiring mode, which is to auto-wire by name (i.e. to look for beans defined in Spring with the same name as your bean property), then you'll also need a setting for this in your [webwork.properties](#):

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

Options for this setting are:

name	Auto-wire by matching the name of the bean in Spring with the name of the property in your action. This is the default
type	Auto-wire by looking for a bean registered with Spring of the same type as the property in your action. This requires you to have only one bean of this type registered with Spring
auto	Spring will attempt to auto-detect the best method for auto-wiring your action
constructor	Spring will auto-wire the parameters of the bean's constructor

At this point, all objects will at least try to get created by Spring. If they cannot be created by Spring, then WebWork will create the object itself. Next, you'll need to turn on the Spring listener in web.xml:

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



More ApplicationContext configuration files needed?

Since the Spring integration uses a standard Listener, it can be configured to support configuration files other than applicationContext.xml.

Adding the following to your web.xml will cause Spring's ApplicationContext to be initialized from all files matching the given pattern:

```
<!-- 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>
```

See the Spring documentation for a full description of this parameter.

Sample Spring Configuration

At this point, you can add the standard Spring configuration at **WEB-INF/applicationContext.xml**. An example of this configuration is:

```
<?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>
```

Switching from Builtin IoC to Spring

Switching is quite easy. Spring setup is done as described above. To complete migration, you will have to

1. transfer your configured components from components.xml to applicationContext.xml appropriately. You can safely delete components.xml afterwards.
2. remove the [Component Interceptor](#) from your interceptor stack in [xwork.xml](#). Although it does not hurt to leave it there, it is simply redundant from now on.



Session Scope & Spring

Spring <= 1.3 does not support session scoped components. Spring 2.0 release will add support for this, and tests with Spring 2.0M3 (beta) are reported to work quite well. Please refer to [Spring Session Components Workarounds](#) to read more about this topic.

Initializing Actions from Spring

Normally, in xwork.xml you specify the class for each action. When using the SpringObjectFactory (configured as shown above) WebWork will ask Spring to create the action and wire up dependencies as specified by the default auto-wire behavior. The SpringObjectFactory will also apply all bean post

processors to do things like proxy your action for transactions, security, etc. which Spring can automatically determine without explicit configuration. For most usages, this should be all you need for configuring your actions to have services and dependencies applied.

-  We **strongly** recommend that you find declarative ways of letting Spring know what to provide for your actions. This includes making your beans able to be autowired by either naming your dependent properties on your action the same as the bean defined in Spring which should be provided (to allow for name-based autowiring), or using autowire-by-type and only having one of the required type registered with Spring. It also can include using JDK5 annotations to declare transactional and security requirements rather than having to explicitly set up proxies in your Spring configuration. If you can find ways to let Spring know what it needs to do for your action without needing any explicit configuration in the Spring *applicationContext.xml*, then you won't have to maintain this configuration in both places.

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.

Your *xwork.xml* file would then have the action class attributes changed, leaving it like this:

```
<!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>
```

Where you have a Spring bean defined in your **applicationContext.xml** named "bar". Note that the *com.acme.Foo* action did not need to be changed, because it can be autowired.

A typical spring configuration for bar could look as following.

```
<?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>
```

or if you are using Spring version 2.0,

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE beans PUBLIC "-//SPRING//DTD BEAN 2.0//EN"
"http://www.springframework.org/dtd/spring-beans-2.0.dtd">
<beans default-autowire="autodetect">
    <bean id="bar" class="com.my.BarClass" scope="prototype"/>
    ...
</beans>
```



Note the id attribute in the spring configuration corresponds to the class attribute in the xwork configuration. Also note that in the spring configuration, the singleton attribute is set to false. This would generally be the case that is desired as Webwork creates a new action class upon each request. Hence when Spring integration is used, this would be the desired behaviour. Making Springs singleton attribute false would allow this.

Remember: **this is not required**. This is only needed if you wish to override the default behavior when the action is created in WebWork by decorating it with Spring-enabled interceptors and IoC that cannot be automatically determined by Spring. Keep in mind that WebWork's Spring integration will do standard IoC, using whatever auto-wiring you specify, even if you don't explicitly map each action in Spring. So typically you don't need to do this, but it is good to know how this can be done if you need to.

Spring AOP and WebWork Actions

By default Spring relies on you using interface based proxying. As you are likely to not have interfaces for your actions you will need to use CGLib for proxying.

Turn on class proxying in your Spring context.

```
<aop:config proxy-target-class="true" />
```

Also ensure CGLib is then on your classpath. This will apply CGLib proxying to **all** classes. Consider delegating to a service layer.

Other Spring Integration

This page last changed on Sep 21, 2005 by [plightbo](#).



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 be 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 Jan 23, 2006 by [cameronbraid](#).

Motivation

Spring currently does not support session scoped beans/components out of the box. You can decide between singleton or prototype lifecycle, but you cannot have your beans bound to the session lifecycle of web applications. There are plans for integrating such a feature in the Spring 2.0 release.

We will try to point out some possible workarounds for your WebWork based applications. First we look at the general solutions found among the Spring community, dealing with HttpSession and all that. After that we will discuss the special conditions and requirements found in XWork/WebWork and how that might affect possible solutions. We will show some XWork/WebWork specific solutions for the given problem.



The first milestone of Spring 2.0 (formerly 1.3) will be released the second week of December 2005. It is confirmed that it does contain Session Scope components using the Proxy (CGLIB or JDK) approach.

General Solutions for Webapplications

The Spring 2.0 Way

Interface21 added support for session (and request) scoped beans in Spring 2.0. This approach creates a CGLIB or JDK Dynamic proxy of the session scoped bean using the org.springframework.aop.target.scope.ScopedProxyFactoryBean and setting the scopeMap to org.springframework.web.context.scope.SessionScopeMap.

Since the jars are backwards compatible simply build Spring and replace the jars shipped with WebWork. (Spring 2.0 M1 should be out by the time you read this.).

There are 2 ways to set this up depending upon whether or not XML simplification is used. The first method uses the traditional bean definitions and is useful to understand what is happening under the covers.

A modified applicationContext.xml for the shopping cart example using the traditional XML DTD is below.

```
<?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>

```

A modified applicationContext.xml for the shopping cart example using the XML simplification is below.

```

<?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/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>

```

You will also need to modify the web.xml to include the following 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>

```

Custom TargetSource with a ServletFilter

A quite "clean" solution for web applications in general can be found at [JA-SIG](#). The solution is well documented and can be found [here](#). Below you will find a WebWork adoption of this solution.

XWork/WebWork specific solutions

Preface

WebWork is based on XWork, and XWork is not tied to the web layer. So when dealing with session scoped objects, WW users might want to use XWork's session abstraction features to keep their application independent from the web context. This is why we will discuss some XW/WW specific solutions

below.

Custom TargetSource, the WebWork way

Here is a modified version of the TargetSource solution pointed out above that integrates with the existing WebWork session and doesn't require an additional filter or listener. Usage is pretty much the same, create an interface for your object and make sure that you always use that interface and not the underlying implementation or autowiring will fail. You can find more information on how to make this work by looking at the [WebWorkTargetSource Shopping Cart Example](#).

```
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 collaboration 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@unicorn.net">edalquist@unicorn.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
     */
    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.newInstance();
        }

        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.newInstance();
        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;
        }
    }
}

```

Customized ApplicationContext Implementation

TODO: Document

Customized WW/XW ObjectFactory

TODO: Document

Session backed Bean Factory

The idea is to simply create a retrieve-or-create 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;
    }
}

```

Example applicationContext setup (note that the session scoped bean has to be setup with 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>

```

Example action use:

```

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;
        }
    }
    ...
}

```

For well known session scoped components, you might get more convenience by subclassing 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);
    }
}

```

Again example applicationContext setup:

```

<?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>
```

Example action use:

```
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;
        }
    }
    ...
}
```

As said, the solution is very simple. You will get no ties to web layer, and the configuration is really simple, there is no need for proxy definitions in applicationContext.xml etc.

The main disadvantage is that you will not be able to wire session scoped beans directly into your actions, you will have to use the indirection via the session backed bean factory. And, as always when dealing with XWork session abstraction, you have to take care for a action context to be setup.

Auto proxied Session backed Component Factory

Does anyone have an implementation of this? (Eric Molitor)

The intention was a bit different for this one, so I tried to clarify headings. Nice idea, though ... (Rene Gielen).

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 approriate value out of the Session and calls the Components setter method.

TODO: Document, create example

WebWorkTargetSource Shopping Cart Example

This page last changed on Nov 30, 2005 by [emolitor](#).

WebWorkTargetSource Shopping Cart Example

Here is a modified version of the shopping cart example which uses the WebWorkTargetSource. Its a quick hack to show how the WebWorkTargetSource and not as a complete solution or template for usage. If my documentation is unclear (probable) or none of this makes sense (quite possible too) then just replace the existing versions of DefaultCart.java and applicationContext.xml with these versions and fire up the example.

DefaultCart Modifications

Two modifications to DefaultCart.java are necessary in order to make the autowiring work. When spring goes to look for beans to autowire it will see two ShoppingCarts and barf as for autowiring to work it needs to see only one. To avoid this the DefaultCart has been modified to not implement the ShoppingCart interface. However there is a fun inner class and inner interface that makes this change a bit more complex. In order to make the DefaultCart compile (and still work) all references to CartEntry need to be changed to 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 Modifications

In order to get a session specific shopping cart we need to modify the actionContext to call our WebWorkTargetSource. We do this by using a ProxyFactory which creates an object based on our interface (ShoppingCart) and uses the targetSource to invoke our custom TargetSource (WebWorkTargetSource). WebWorkTargetSource however needs to know the underlying implementation in order to fetch and create new instances. We pass a reference to the new shoppingCartTarget bean definition which just references our new DefaultCart. In order to keep things from getting confused we're set both beans to autowire by name.

```

<?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>

```

Spring Webflow

This page last changed on Jan 22, 2007 by [tschneider22](#).

The [WebworkWebflow](#) project integrates [Spring Webflow](#) with Webwork

Project Team

- Tom Schneider

Requirements

- Spring 2.0+
- Spring Webflow 1.0+
- Webwork 2.2.4+
- WebworkWebflow 1.0.0+

Prerequisite

For background information on the core Spring Webflow concepts, visit <http://www.ervacon.com/products/swf/intro/>.

Step 1 - Create your flow definition xml

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

    <start-state idref="ageEnter" />

    <view-state id="ageEnter" view="ageEnter">
        <transition on="submit" to="AgeSave" />
    </view-state>

    <view-state id="ageEnterJSP" view="ageEnter">
        <transition on="input" to="enterAgeJSP" />
        <transition on="submit" to="AgeSave" />
    </view-state>

    <action-state id="AgeSave">
        <action bean="webworkFlowAdapter"/>
        <transition on="input" to="enterAgeJSP" />
        <transition on="success" to="calcRate" />
    </action-state>

    <view-state id="calcRate" view="calcRate">
        <transition on="finish" to="finish" />
    </view-state>

    <end-state id="finish" view="finish"/>
</flow>
```

Step 2 - Configure Spring's applicationContext.xml

```

<?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:flow="http://www.springframework.org/schema/webflow-config"
    xsi:schemaLocation="
        http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.0.xsd
        http://www.springframework.org/schema/webflow-config
        http://www.springframework.org/schema/webflow-config/spring-webflow-config-1.0.xsd">

    <!-- Launches new flow executions and resumes existing executions. -->
    <flow:executor id="flowExecutor" registry-ref="flowRegistry">
        <flow:execution-attributes>
            <flow:alwaysRedirectOnPause value="false"/>
        </flow:execution-attributes>
    </flow:executor>

    <!-- Creates the registry of flow definitions for this application -->
    <flow:registry id="flowRegistry">
        <flow:location path="/WEB-INF/flows/**-flow.xml"/>
    </flow:registry>

    <bean id="webworkFlowAdapter"
        class="com.googlecode.webworkwebflow.WebworkFlowAdapter"></bean>
</beans>

```

The WebworkFlowAdapter allows a webwork action to execute a webflow action-state. The WebworkFlowAdapter uses the id of the action state as the name of webwork action to execute. The *alwaysRedirectOnPause* is disabled because whether to redirect or not is easier to control in the webwork configuration files.

Step 3 - Add SWF interceptors to webwork's xwork.xml configuration file

```

<interceptors>
    <interceptor name="sessionFlowExecKey"
        class="com.googlecode.webworkwebflow.SessionFlowExecKeyInterceptor"/>
    <interceptor name="flowScope" class="com.googlecode.webworkwebflow.FlowScopeInterceptor">
        <param name="flowScope">age</param>
    </interceptor>
</interceptors>

```

The SessionFlowExecKeyInterceptor puts the flow execution key in the session rather than having it as a hidden field on the form that submitted back. The FlowScopeInterceptor takes a common separated list of variables and binds them to flow scope. Before an action executes, this interceptor looks in flow scope and populates the corresponding properties on the webwork action. After the action has executed, the variables are retrieved from the action and the updated values are put back into flow scope.

Step 4 - Configure the FlowAction so Spring Webflows can be executed

```

<action name="FlowAction" class="com.googlecode.webworkwebflow.FlowAction">
    <interceptor-ref name="sessionFlowExecKey" />
    <interceptor-ref name="defaultStack" />
    <param name="flowId">rating-flow</param>
    <result name="ageEnter" type="redirect">
        AgeEnter.action
    </result>
    <result name="ageEnterJSP">/example/enterage.jsp</result>
    <result name="calcRate" type="redirect">
        CalcRate.action
    </result>
    <result name="finish">/example/finished.jsp</result>

```

```
</action>
```

For each view-state defined in the flow xml definition, there should be a corresponding result entry in the FlowAction definition. (Or a global result for the view state)

Step 5 - Access the flow

The flow can now be launched by accessing the FlowAction.

Tiles

This page last changed on Feb 27, 2006 by [phil](#).

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](#))

About WebWork

This page last changed on Jan 03, 2007 by [phil](#).

App Servers

This page last changed on Oct 24, 2006 by [phil](#).

The following is a list of application servers WebWork is known to work with, and any potential workarounds necessary for full functionality:

- [WebLogic](#)
- [WebLogic 6.1](#)
- [SunOne 7.0](#)
- [WebSphere](#)
- JRun
- Jetty
- Tomcat/JBoss
- Resin
- Orion
- OC4J
- [Glassfish](#)

Glassfish

This page last changed on Oct 24, 2006 by [phil](#).

From the forum:

- <http://forums.opensymphony.com/thread.jspa?threadID=47061&tstart=0>

Robert Watkins: "Anyone who is trying to deploy multiple WebWork-based web applications in a single EAR file to Glassfish or Sun App Server 9 should probably check out this bug:

https://glassfish.dev.java.net/issues/show_bug.cgi?id=1351

Basically, with the default configuration, Enterprise apps would share the webwork JARs (particularly xwork) which means the static configuration gets "shared" to. Probably not what you want, and definitely not what I want.

It appears that disabling classloader delegation for the web app removes the shared state, allowing you to work around the problem at the cost of increased memory usage."

SunOne 7.0

This page last changed on Mar 23, 2006 by [plightbo](#).

You need to grant permissions to WebWork:

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

or more specifically,

- Give Write Permissions to `java.util.PropertyPermission`.
- Add `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 Nov 20, 2004 by [mcampbell](#).

```
> 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/.

WebLogic 6.1

This page last changed on Jun 06, 2005 by [plightbo](#).

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 initialization.

However, the WebWork code cannot be modified 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.

WebSphere

This page last changed on Aug 23, 2006 by [rainerh](#).

Websphere

Websphere 5.1

- <http://forums.opensymphony.com/thread.jspa?threadID=26068&tstart=0>

"Summary: Make sure to install fix pack 1".

Websphere 6.0

- <http://forums.opensymphony.com/thread.jspa?threadID=26068&messageID=68992#68992>

Eivind Waaler wrote:

Figured out a "solution". It seems I need to get the web application loaded on startup. To achieve this I just added the old ServletDispatcher with a load-on-startup setting, without any servlet mapping. In web.xml, simply add these lines:

```
...
<servlet>
    <servlet-name>dummyaction</servlet-name>
    <servlet-class>com.opensymphony.webwork.dispatcher.ServletDispatcher</servlet-class>
    <load-on-startup>1</load-on-startup>
</servlet>
...
```

Hope this can help others with the same problem. I've been using latest WebWork (2.2.2) and WebSphere (6.0.2.11) as of writing.

Articles and press

This page last changed on Oct 20, 2006 by [phil](#).

WebWork is a very popular framework and community. As such, there are many articles, presentations, and books about WebWork. Here is just a sample.

Books

- [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

Presentations

- 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](#)
- [Seven Simple Ways to Use AppFuse](#)

Articles

- [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
- [Migrating Applications to Struts2 \(Part 1\)](#) - September, 2006
- [Migrating Applications to Struts2 \(Part 2\)](#) - October, 2006

Blogs

The official WebWork Blog can be found [here](#).

Additionally, the blogs of the developers of WebWork may provide some useful information:

- [Blogbody](#) - Patrick Lightbody
- [Jason Carreira](#)

Blog entries:

- [A summary of what is new in WebWork 2.0](#) - Mike Cannon-Brookes
- [WebWork Performance Tips](#) - AgileBlog

Strutting the OpenSymphony way

This page last changed on Jan 03, 2007 by [phil](#).

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:XWork1 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 componentize 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](#).

Companies that provide WebWork support

This page last changed on Jan 29, 2007 by [phil](#).

Are you looking for help with your WebWork project? Perhaps you are looking to have one WebWork expert on your team to provide mentoring and coaching. Here is a list of professionals or companies that provide WebWork support and or development.

Name	Contact	Location	Type
AgileEdge Software	Mike Porter - mike at agileedge dot com	Seattle - But will travel w/ T&L	
Aixcept	Rainer Hermanns - hermanns at aixcept dot de	Aachen, Germany. Also for projects in: Belgium/Netherlands/France	
From Down & Around, Inc	Ian Roughley - ian at fdar dot com	Based out of Boston, MA - will travel	
Partnet	James House - jhouse at part dot net	Salt Lake City	

If you think you fall into the category of a WebWork expert, feel free to add your details in the table above.

Comparison to other web frameworks

This page last changed on Feb 22, 2006 by [phil](#).

- [Comparison to Struts](#)
- [Comparison to JSF](#)
- [Comparison to Tapestry](#)
- [Comparison to Spring MVC](#)
- [Comparison to Ruby on Rails](#)

[Matt Raible](#) wrote (mid 2005) an interesting whitepaper where he compared various web frameworks.

You can view the PDF here:

<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 Oct 30, 2005 by [plightbo](#).

TODO: a brief write-up comparing WebWork to JSF

Comparison to Ruby on Rails

This page last changed on Oct 30, 2005 by [plightbo](#).

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 Oct 30, 2005 by [plightbo](#).

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

Comparison to Struts

This page last changed on Jan 07, 2006 by [plightbo](#).



As of December, 2005 the Struts and WebWork teams have agreed to combine their efforts to create Struts Action 2.0. Struts Action 2.0 will be based on WebWork 2.2. Therefore this comparison is less relevant as the two communities cooperate more.

Feature Comparison

Feature	Struts	WebWork 1.x	WebWork 2.x
Action classes	Struts requires Action classes to extend an Abstract base class. This shows a common problem in Struts of programming to abstract classes instead of interfaces.	Action classes must implement the webwork.Action Interface. There are other Interfaces which can be implemented for other services, such as storing error messages, getting localized texts, etc. The ActionSupport class implements many of these Interfaces and can act as a base class. WebWork is all written to Interfaces, which allows for plugging in your own implementations.	An Action must implement the com.opensymphony.xwork.Action Interface, with a series of other Interfaces for other services, like in WebWork 1.x. WebWork2 has its own ActionSupport to implement these Interfaces.
Threading Model	Struts Actions must be thread-safe because there will only be one instance to handle all requests. This places restrictions on what can be done with Struts Actions as any resources held must be thread-safe or access to them must be synchronized.	WebWork Actions are instantiated for each request, so there are no thread-safety issues. In practice, Servlet containers generate many throw-away objects per request, and one more Object does not prove to be a problem for performance or garbage collection.	ditto
Servlet Dependency	Struts Actions have dependencies on Servlets because they get the HttpServletRequest and HttpServletResponse (not HttpServletRequest and HttpServletResponse, I've been told) when they are executed. This tie to Servlets (although not Http*) is a defacto tie to the	WebWork Actions are not tied to the web or any container. WebWork actions CAN choose to access the request and response from the ActionContext, but it is not required and should be done only when ABSOLUTELY neccessary	ditto

	<p>Servlet container, which is Web. an unneeded dependency.</p> <p>Servlets may be used outside a Web context, but it's not a good fit for JMS, for instance.</p>	
Testability	<p>Many strategies have sprung up around testing Struts applications, but the major hurdle is the fact that Struts Actions are so tightly tied to the web (receiving a Request and Response object). This often leads people to test Struts Actions inside a container, which is both slow and NOT UNIT TESTING. There is a Junit extension : Struts TestCase (http://strutstestcase.sourceforge.net/)</p>	<p>WebWork actions can be tested by instantiating your action, setting the properties, and executing them</p> <p>ditto, but the emphasis on Inversion of Control makes testing even simpler, as you can just set a Mock implementation of your services into your Action for testing, instead of having to set up service registries or static singletons</p>
FormBeans	<p>Struts requires the use of FormBeans for every form, necessitating either a lot of extra classes or the use of DynaBeans, which are really just a workaround for the limitation of requiring FormBeans</p>	<p>WebWork 1.x allows you to have all of your properties directly accessible on your Action as regular Javabeans properties, including rich Object types which can have their own properties which can be accessed from the web page. WebWork also allows the FormBean pattern, as discussed in "WW1:Populate Form Bean and access its value"</p>
Expression Language	<p>Struts 1.1 integrates with JSTL, so it uses the JSTL EL. This EL has basic object graph traversal, but relatively weak collection and indexed property support.</p>	<p>WebWork 1.x has its own Expression language which is built for accessing the ValueStack. Collection and indexed property support are basics but good. WebWork can also work with JSTL</p> <p>WebWork uses Ognl which is a VERY powerful expression language, with additions for accessing the value stack. Ognl supports very powerful collection and indexed property support. Ognl also supports powerful features like projections (calling the same method on each member of a collection and building a new collection of the results), selections (filtering a collection with a selector expression to</p>

Binding values into views	Struts uses the standard JSP mechanism for binding objects into the page context for access, which tightly couples your view to the form beans being rendered	WebWork sets up a ValueStack which the WebWork taglibs access to dynamically find values very flexibly without tightly coupling your view to the types it is rendering. This allows you to reuse views across a range of types which have the same properties.	return a subset), list construction, and lambda expressions (simple functions which can be reused). Ognl also allows access to static methods, static fields, and constructors of classes. WebWork2 may also use JSTL as mentioned in WW1:Using JSTL seamlessly with WebWork ditto
Type Conversion	Struts FormBeans properties are usually all Strings. Struts uses Commons-Beanutils for type conversion. Converters are per-class, and not configurable per instance. Getting a meaningful type conversion error out and displaying it to the user can be difficult.	WebWork 1.x uses PropertyEditors for type conversion. PropertyEditors are per type and not settable per Action, but field error messages are added to the field error map in the Action to be automatically displayed to the user with the field.	WebWork2 uses Ognl for type conversion with added converters provided for all basic types. Type converters default to these converters, but type conversion can be specified per field per class. Type conversion errors also have a default error message but can be set per field per class using the localization mechanism in WW2 and will be set into the field error messages of the Action.
Modular Before & After Processing	Class hierarchies of base Actions must be built up to do processing before and after delegating to the Action classes, which can lead deep class hierarchies and limitations due to the inability to have multiple inheritance WW:Comparison to Struts#1	Class hierarchies	WebWork 2 allows you to modularize before and after processing in Interceptors. Interceptors can be applied dynamically via the configuration without any coupling between the Action classes and the Interceptors.
Validation	Struts calls validate() on the FormBean. Struts	WebWork1.x calls the validate() method on	WebWork can use the validate() method of

	<p>users often use Commons Actions, which can either Validation for validation. I do programmatic don't know a lot about this, so I'll put some questions here: Because FormBean properties are usually Strings, some types of validations must either be duplicated (checking type conversion) or cannot be done?</p> <p>Can Commons Validation have different validation contexts for the same class? (I've been told yes, so that's a good thing)</p> <p>Can Commons Validation chain to validations on sub-objects, using the validations defined for that object properties class?</p>	<p>WebWork and Struts and / or use the Validation framework, which is activated using an XWork Interceptor. The Xwork Validation Framework allows you to define validations in an XML format with default validations for a class and custom validations added for different validation contexts. The Xwork Validation Framework is enabled via an Interceptor and is therefore completely decoupled from your Action class. The Xwork Validation Framework also allows you to chain the validation process down into sub-properties using the VisitorFieldValidator which will use the validations defined for the properties class type and the validation context.</p> <p>The interceptor stacks in WebWork 2 are hugely powerful in this regard. All aspects of Action setup have been moved into Interceptor implementations (ie setting parameters from the web, validation etc), so you can control on a per action basis the order in which they are performed. For example you might want your IOC framework to setup the action before the parameters are set from the request or vice versa - you can thusly control this on a per package or per action basis with interceptor stacks.</p>
Control Of Action Execution	<p>As far as I know Struts sets up the Action object for you, and you have very little control over the order of operations. To change them I think ? you need to write your own Servlet to handle dispatching as you want</p>	<p>The ActionFactory chain controls the order in which an Action is constructed and initialised, but this requires writing a class</p>

References

- <http://www.mail-archive.com/opensymphony-webwork@lists.sourceforge.net/msg00995.html> - compares Struts development to WebWork 1.x development from the point of view of a Struts developer who switched to WebWork
- <http://www.mail-archive.com/opensymphony-webwork@lists.sourceforge.net/msg04700.html> - Kind of the first draft of this comparison

Footnotes

1. Some Struts users have built the beginnings of an Interceptor framework for Struts (<http://struts.sourceforge.net/saif/>). It currently has some serious limitations (no "around" processing, just before and after) and is not part of the main Struts project

Comparison to Tapestry

This page last changed on Oct 30, 2005 by [plightbo](#).

TODO: a brief write-up comparing WebWork to Tapestry

Press Releases

This page last changed on Jan 03, 2007 by [phil](#).

Here you will find all the press releases for WebWork. Each press release should include the [About](#) page at the top of it, which you don't need to worry about as long as you use the standard Press Release template.

- [2.1.7 Press Release](#)
- [2.1.6 Press Release](#)
- [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 Jun 23, 2004 by [plightbo](#).

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

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

2.1.1 Press Release

This page last changed on Aug 27, 2004 by [plightbo](#).

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.

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities

of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

2.1.2 Press Release

This page last changed on Oct 15, 2004 by [plightbo](#).

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.

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

2.1.3 Press Release

This page last changed on Oct 15, 2004 by [plightbo](#).

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>

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

2.1.4 Press Release

This page last changed on Oct 18, 2004 by [plightbo](#).

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 be 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>

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

2.1.5 Press Release

This page last changed on Oct 23, 2004 by [plightbo](#).

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.

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

2.1.6 Press Release

This page last changed on Nov 14, 2004 by [plightbo](#).

WebWork 2.1.6 Released

The [OpenSymphony](#) group is proud to announce the release of WebWork 2.1.6.

- WebWork: <http://www.opensymphony.com/webwork/>
- Release notes and changes:
<http://www.opensymphony.com/webwork/wikidocs/Release%20Notes%20-%202.1.6.html>
- Documentation: <http://www.opensymphony.com/webwork/wikidocs/Documentation.html>
- Download: <https://webwork.dev.java.net/files/documents/693/8838/webwork-2.1.6.zip>

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

2.1.7 Press Release

This page last changed on Dec 14, 2004 by [plightbo](#).

WebWork 2.1.7 Released

The [OpenSymphony](#) group is proud to announce the release of WebWork 2.1.7. This release is focused on minor improvements around the UI framework, including a new and powerful client-side validation framework currently in prototype phase.

- WebWork: <http://www.opensymphony.com/webwork>
- Release notes and changes:
<http://www.opensymphony.com/webwork/wikidocs/WebWork%202.1.7.html>
- Documentation: <http://www.opensymphony.com/webwork/wikidocs/Documentation.html>
- Download: <https://webwork.dev.java.net/files/documents/693/9723/webwork-2.1.7.zip>

Special Thanks

The WebWork team would like to recognize the following people for their help with bug reports, user support, documentation, and patches:

- Mathias Bogaert for fixing so many JIRA issues
- Casey Moyaert for managing most of the documentation and continuing to make it better

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually

associated with building web-based user interfaces.

- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

About

This page last changed on Jun 23, 2004 by [plightbo](#).

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

Previous releases

This page last changed on Mar 25, 2007 by [phil](#).

- Release Notes
 - [WebWork 2.2.5](#)
 - [WebWork 2.2.4](#)
 - [WebWork 2.2.3](#)
 - [WebWork 2.2.2](#)
 - [WebWork 2.2.1](#)
 - [WebWork 2.2](#)

Old format of release notes and upgrade guides

- Release Notes
 - [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](#)
- Upgrading from previous versions
 - [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 Dec 14, 2004 by [plightbo](#).

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 <code>webwork.ui.templateDir</code> configuration property has been broken into <code>webwork.ui.templateDir</code> and <code>webwork.ui.theme</code>	<code>webwork.ui.templateDir = /webwork/mytheme</code>	<code>webwork.ui.templateDir = /webwork</code> <code>webwork.ui.theme = mytheme</code>
2.0	"namespace" attribute of the <code>ww:action</code> tag is now evaluated; those upgrading from 2.0 will need to place single quotes around the attribute value	<code><ww:action namespace="/foo" .../></code>	<code><ww:action namespace="'/foo'" .../></code>
2.0, but not 1.x	theme and template	<code><ww:xxxx</code>	<code><ww:xxxx</code>

	attributes in UI tags have changed are now evaluated; those upgrading from 2.0 will need to place single quotes around the attribute value	theme="/template/foo" template="bar.vm"/>	theme="foo" template="bar.vm"/>
1.x, 2.0	label UI tag evaluates the value attribute now instead of the name attribute	<ww:label name="Foo"/>	<ww:label value="Foo"/>

Changelog

OpenSymphony JIRA (25 issues)		
T	Key	Summary
	WW-592	Upgrade commons-logging
	WW-560	SessionMap holds on to requests when it doesn'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't work after latest ActionConfig refactoring
	WW-519	ActionTag should evaluate namespace attribute
	WW-518	Label attribute shouldn'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	JavaScript-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 Dec 14, 2004 by [plightbo](#).

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'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'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'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 overridden
	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's short circuit behaviour
	XW-179	Optimise OgnlUtil.copy method
	XW-172	XWorkBasicConverter doesn't care about the current locale

Release Notes - 2.1.2

This page last changed on Dec 14, 2004 by [plightbo](#).

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 shipped 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 'name' 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'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](#)
[WW-485](#)

[Stream Result Type](#)
[Add docs for WebWork2 tags](#)

Release Notes - 2.1.3

This page last changed on Dec 14, 2004 by [plightbo](#).

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 Dec 14, 2004 by [plightbo](#).

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 because 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 Dec 14, 2004 by [plightbo](#).

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't initialize VelocityManager
	WW-660	xhtml's checkbox.vm vertical alignment
	WW-658	JSP Tags do not support onFocus, onBlur js handlers
	WW-653	url taglib does not support 'page' 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

WW-526	JasperReportsResult.java
WW-479	Velocity using include ignores character encoding
WW-425	getValueClassType() in ComboboxTag returns Boolean.class
WW-392	Refactoring to decouple the UI Tag dependency on velocity
WW-223	Update doubleselect tag
WW-161	Multipart & SaveDir
	ServletDispatcher could be a lot simpler to extend

Release Notes - 2.1.6

This page last changed on Dec 14, 2004 by [plightbo](#).

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 subclass TextFieldTag

Upgrading from 1.4

This page last changed on Dec 11, 2005 by [plightbo](#).

Package changes

Webwork1.x was separated 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 automatically 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 acquired 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 Apr 12, 2005 by [plightbo](#).

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 current object.	No longer used.
Reference a static variable	@com.fully.qualified.class.name.TheStaticClass@STATIC_ATTRIBUTE_NAME	

Originally written by Jay Bose and sent to the mailing list

Upgrading from 2.0

This page last changed on Dec 14, 2004 by [plightbo](#).

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 Dec 14, 2004 by [plightbo](#).

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 Dec 14, 2004 by [plightbo](#).

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 Dec 14, 2004 by [plightbo](#).

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 Dec 14, 2004 by [plightbo](#).

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 Dec 14, 2004 by [plightbo](#).

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 Dec 14, 2004 by [plightbo](#).

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 Dec 11, 2005 by [plightbo](#).

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't check the allowed files'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 FreemarkerWebworkUtil
	WW-686	JasperReport result - Content-Disposition management
	WW-684	Default action class and result name.
	WW-681	getText() doesn'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 BigDecimal
WW-351	Forwarding unknown tag attributes

WebWork 2.2

This page last changed on Jan 09, 2006 by [plightbo](#).

WebWork 2.2 Release Notes

Key Changes

Productivity enhancements

Tools

- Fully functional WebFlow support for JSP, FreeMarker, and Velocity
- [QuickStart](#): simple embedded application server for getting started immediately

Documentation

- Improved documentation, including detailed information for every interceptor
- Totally new example application: now the example application is a set of tutorials and only teaches best practices of WebWork, rather than every single feature.

Enhanced framework feedback

- Better and more intelligent error reporting
- "Developer" mode where inline errors are displayed when possible
- Common interceptor stack issues, such as validation+workflow with no "input" result, are now reported in a more obvious way

Other

- Deprecated WebWork IoC container in favor of Spring
- Built-in support for Spring
- Official support for wizards/workflows using the [Scope Interceptor](#) and a pre-release of [Continuations](#)
- Removed support for WebWork 1.x migration jar

User interface improvements

UI tag overhaul

- FreeMarker is now the default UI tag implementation
- Refactored UI tag base classes such that they are no longer tied to JSP
- New native Velocity and FreeMarker UI tag support, built on top of new base classes
- UI tags now use "altSyntax" (available since 2.1.4) as the default syntax (the 2.0 - 2.1 syntax is deprecated but still available)

- New [Head](#) tag that links the correct CSS and JavaScript files for each theme

Velocity Support Improvements

- Upgraded support to velocity 1.4
- webwork.velocity.contexts now chains contexts on each request, i.e. contexts do not need to be thread-safe

AJAX support

- Official support for client side validation using DWR
- New tabbed panel widget
- Built in support for Dojo widgets

Result changes

- Velocity and FreeMarker Servlets are now deprecated in favor of direct results

Other

- Easy way to invoke different action name or command, making forms with multiple buttons easy to use
- Initial JSR168 integration

Core API changes

Type conversion

- Typing support in Maps, Sets, and Lists is now supported even when the collection is not null
- Map type conversion support for keys and values
- Support for Java 5 generics and annotations for Collections and enums

Other

- Improved exception handling, with support for exception-to-result mapping in xwork.xml
- Parameters interceptor updated to let you include and/or exclude certain parameters, thereby providing a simple way to secure what data can be changed from the web

Migration Notes

WebWork 2.2 is the most significant release since the 2.0 release two years ago. There are some significant changes, deprecated items, and various issues to be aware of when upgrading or if you're just curious what is new. **Please see the [WebWork 2.2 Migration Notes](#) for more info.**

Changelog

For a complete list of all the changes, please refer to the [complete changelog](#)

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'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'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't accept null of CompoundRoot element
	WW-1048	RemoteCallUIBean uses UrlHelper

WW-1047	to build url for href attribute breaks <ww:a /> tag
WW-1046	Fix broken locale files of jscalendar 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't work with new dojo
WW-1030	Loading Freemarker templates with the ActionContext'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'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.WebWorkP
WW-1014	for Portlet Application webwork 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'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-978	ww:sort ww:generator and ww:subset tags are broken
WW-977	Input and Output streams not closed in StreamResult
WW-976	WW Portlets can'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'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'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.Resource
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

WW-900	Interceptor Documentation 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'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'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 delimiter
WW-811	StreamResult doesn'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'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 & 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'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
WW-742	Default Action Method
WW-740	wrap attribute in textarea.vm in 'simple' template
WW-739	UI tag checkboxlist should have html scripting events attribute
WW-738	Action tag TLD missing
WW-737	ignoreContextParams
WW-736	Use ObjectFactory in velocityManager for setting webwork.velocity.contexts
WW-735	JasperReports examples in the webwork-example app not working
WW-730	XmlHttpRequest validation breaks on IE
WW-729	\$webwork.evaluate() throws NPE
WW-725	2.1.7 text taglib is not compatible with old version
WW-721	UrlTag always includes parameters unless includeParams is explicitly set to false
WW-720	The include tag doesn't cleanup after itself
WW-719	<code><ww:subset/></code> problem
WW-717	Add content-length to StreamResult
WW-713	Session object not auto-shared with Velocity context
WW-712	The "label" tags is missing the "for" attribute
WW-709	The <ww:url> tag does not correctly construct the URL when the scheme attribute is set and the scheme changes.
WW-708	Update config browser to work with the new syntax
WW-707	Allow specifying a text for empty option on the select tag
WW-700	AbstractListTag classes throw NPE for null value
WW-697	Velocity templates in UTF-8 still do not work
WW-682	Validation lacking logging of missing field.
WW-679	Regex Field Validator change
	webwork.ui.templateDir=otherDir, tag_class, doEndTag: Unable to find resource '/template/simple/scripting-events.vm' error.
	The "Component" UI tag can'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'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'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 & 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't honored for command driven

	<u>actions</u>
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 'name' 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-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 Jan 07, 2006 by [plightbo](#).

This document covers a step-by-step guide for upgrading to WebWork 2.2 from 2.1.x, as well as a list of the key individual changes for reference.

Upgrade Guide

1. Get the [latest 2.2 release](#)
2. Check out the [dependencies](#) to see what the required libraries are. One change of note is the dependency on Rife-Continuations. Click through the tabs for the dependencies for different usage profiles. If you use FreeMarker for instance, click on that tab to see those dependencies. Note that if you use the JSP tags you are now using FreeMarker by default for the UI component templates.
3. Check the **Individual Changes** section below to see if any of those changes affect your code
4. Update to use the **FilterDispatcher** instead of the **ServletDispatcher**. Check out the [web.xml 2.1.x compatibility](#) page for some compatibility discussions, and see [web.xml](#) for what needs to go in the *web.xml* file.

Deprecated Items

- The ServletDispatcher is now deprecated, please use the FilterDispatcher if possible. See [web.xml 2.1.x compatibility](#) for more information about potential problems you might have when switching to FilterDispatcher.
- The Velocity and FreeMarker servlets are no longer supported. We highly recommend you don't use these servlets but rather the [FreeMarker Result](#) or [Velocity Result](#) directly.
- If you were using JSP tags within Velocity, this is no longer supported and will soon be removed. You can use the instructions explained in [web.xml 2.1.x compatibility](#) to get along for now, but we highly recommend using the new native Velocity tag support provided by WebWork.
- The table tag is now considered deprecated. We may undeprecate it in the future if more time can be invested, but we recommend that you look at alternative options such as [Display Tag](#).
- All support for including actions (using the [include](#) tag or `jsp:include`) is no longer available when using the FilterDispatcher. We recommend you use the [action](#) tag instead.
- The cos and pell file upload parsers are no longer actively maintained and will be deleted soon. We highly recommend you use the Jakarta file upload parser, which is now the default parser.

Deleted Items

- All VoiceXML tags have been removed from WebWork.
- The Velocity-based [Tags](#) have been removed. If you were using or extending these tags (advanced users typically), you can copy them from the `/template/archive` directory in the webwork jar.

Individual Changes

Version	Description	Old Code	New Code
2.1.x	<p>If you implemented your own ObjectFactory or ActionInvocation classes, you will notice that there have been some minor changes to make an "extraContext" Map available for the build* methods. This allows, for instance, access to the Session map during object creation, even before the ActionContext ThreadLocal has been set.</p>	ObjectFactory.getObjectFactory().build(extraContext);	ObjectFactory.getObjectFactory().build(extraContext);
2.0+	<p>If you've used the WebWork base classes for building templated tags, you'll run into the refactoring of the UI tags to use common Component classes as the templated back-end. The tags now use these Component classes, as do Velocity and FreeMarker. This allows Velocity and FreeMarker to use the same UI components directly, without pretending to be a JSP page, but it also means you need to refactor your custom tags to use the new API's</p>	...your code..	See the existing UI tags in the 2.2 source
2.1.x	<p>If you were <i>not</i> using the Alt Syntax, it is now enabled by default. You can either upgrade or change the Tag Syntax</p>	<ww:url value="http://www.yahoo.com"/>	<ww:url value="http://www.yahoo.com"/>
2.1.x	<p>If you are using FreeMarker and your code uses psuedo properties on collections and maps, you need to modify the code to call methods instead.</p>	<code> \${parameters?size} / \${parameters.size?html}</code>	<code> \${parameters.size()} / \${parameters.get("size")?html}</code>
2.1.x	The defaultStack has	<interceptor-ref	<interceptor-ref

	been renamed to the basicStack.	<code>name="defaultStack"/></code>	<code>name="basicStack"/></code>
2.1.x	The completeStack has been renamed to the defaultStack.	<code><interceptor-ref name="completeStack"/></code>	<code><interceptor-ref name="defaultStack"/></code>
2.1.x	The defaultStack (previously the completeStack) is now the default interceptor stack in webwork-default.xml. In addition, this stack now configures the Workflow Interceptor and the Validation Interceptor to not run if the method names are <i>input</i> , <i>back</i> , or <i>cancel</i>	N/A	N/A
2.1.x	The component interceptor has been deprecated (along with all WebWork IOC features) and has been removed from the basicStack and completeStack. You'll need to add it back by hand if you wish to use this deprecated feature.	N/A	N/A
2.0+	The include tag's page attribute has been deprecated since 1.x and is now removed from 2.2. Please use the value attribute.	<code><ww:include page="..."/></code>	<code><ww:include value="..."/></code>
2.0+	The text tag's value0, value1, value2, and value 3attributes have been deprecated since 1.x and are now removed from 2.2. Please use the param tag instead.	<code><ww:text value0="...""/></code>	<code><ww:text><ww:param value="someValue">...</ww:param></code>
2.0+	The session map wrapper (found in ActionContext) has been changed to no longer create sessions for every request. If your application depends on sessions being	N/A	N/A

	automatically created, WebWork 2.2 no longer does that. Instead, you must create the session yourself or the session will be created as soon as a value is put in the session Map.		
2.0+	The VUI tags have been removed from WebWork. They haven't been actively worked on in over 4 years and are not used in the community.	N/A	N/A
2.0+	The URI of the WebWork TLD was change from <i>webwork</i> to <i>/webwork</i> . If you were already using the packaged TLD from <i>webwork.jar</i> , you will have to adjust URI in your JSPs	<%@ taglib uri="webwork" prefix="ww" %>	<%@ taglib uri="/webwork" prefix="ww" %>
2.0+	The default encoding has changed from ISO-8859-1 to UTF-8 . If you wish to continue to use ISO-8859-1, you must change your webwork.properties .	N/A	webwork.i18n.encoding=ISO-8859-1

WebWork 2.2.1

This page last changed on Feb 03, 2006 by [rainerh](#).

WebWork 2.2.1 Release Notes

Key Changes

Portlet Integration

- JSR-168 portlet integration support [Which portal servers are supported](#)
- Improved webwork-portlet sample application
- Added deployment descriptors for various portal servers

Validation

- Improved [Client Side Validation](#)

UI Tags

- New [Tree](#) component
- New UpDown Select Component [updownselect](#)
- Fixed [Debug](#) tag

Tools

- [Quickstart](#) improvement
- Maven support through pom.xml

Misc

- Improved [JSTL](#) support
- More examples within the showcase webapp
- Basic getting started webapp in starter webapp
- 'ant new' task in webapps/build.xml to create your projects project structure [Create a webapp project structure for your web application](#)

Migration Notes

WebWork 2.2.1 is a bugfix release for the 2.2 release a month ago.

If you have used the previous portlet integration code, you should check the portlet-webapp for an example.

The DTD public IDs in xwork.dtd were incorrect in the 1.1. release.

Make sure your DOCTYPE definitions for xwork.xml files look like this:

For xwork 1.1:

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

For xwork 1.1.1:

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

Changelog

For a complete list of all the changes, please refer to the [complete changelog](#)

OpenSymphony JIRA (69 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 librariesxwork.ObjectFactory.getClassInstance callsutil.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'd templates

WW-1127	DevMode impacts pageflow logic
WW-1126	Quickstart won'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 int validator in javascript throws exception if only min or max are set
WW-1117	ww:select tag works only with string variables
WW-1116	Refactor portlet implementation Stack overflow bug fix implemented incorrectly?
WW-1115	XML Field Validation Required not working on Integer or Long values
WW-1114	Debug component should be moved into c.o.w.components package
WW-1112	Debug Tag not working with FTL the javascript in the DHTML response from remove DIV has not yet be evaluated
WW-1111	ognl.OgnlException: templateDir [java.lang.StackOverflowError]
WW-1110	Client-side javascript for basic validators
WW-1109	More Config Browser issues javadoc typo in TokenInterceptor UITag tests need improvements jsessionid params appended to css and js includes
WW-1108	Multipart requests are not cleaned up correctly
WW-1107	jsp view cannot generate onsubmit html tag correctly.
WW-1106	ScopeInterceptor has hardcoded dependencies on HttpServletRequest, etc
WW-1105	Maven Repository at ibiblio does not contain required dependencies for WW/XW
WW-1104	datepicker tag cannot display calendar correctly when browser's locale is zh_CN.
WW-1103	Method Specification for Validation
WW-1102	
WW-1101	
WW-1099	
WW-1098	
WW-1096	
WW-1093	

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 prefixes in DefaultActionMapper will cause ParametersInterceptor problem(OGNL parse error)
WW-1087	make includeParams of URL don'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-978	ww:sort ww:generator and ww:subset tags are broken
WW-958	Config-Browser don'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't override default webwork messages
WW-793	Message with key "webwork.internal.invalid.token" is

[WW-688](#)

[never used](#)

[Refactor TokenInterceptor and
TokenHelper \(and affected classes\)](#)

WebWork 2.2.2

This page last changed on Aug 22, 2006 by [phil](#).

WebWork 2.2.2 Release Notes

Key Changes

Portlets

- Portlet tests added, logging and url fixed
- Added support for Velocity

Validation

- Improved [Client Side Validation](#)

UI and Views

- New and improved components: [Submit](#) now supporting image and html button flavours, [Reset](#), [RichTextEditor](#), [url](#), [Date](#), [Token](#)
- Various components now have tooltip support
- Freemaker and Velocity bugfixes
- Better integration with SiteMesh for Freemaker and JSTL
- Various bug fixes for the xhtml and [ajax theme](#)
- Added support for using Tiles
- Support for new [Plaintext Result](#) type

Tools

- [Quickstart](#) OSX problems fixed

Misc

- Updated and improved documentation, including new [Getting Started](#) pages and [Portlet Tutorial](#)
- Updated test coverage

Migration Notes

WebWork 2.2.2 is a bugfix release, with minor improvements, mostly related to the view layer(s).

Migration from 2.2.2 should be trivial, with one caveat for the [param](#) tag.

WebWork 2.2.2 is the final release under the Opensymphony umbrella. WebWork is now moving to the Apache group and will serve as basis for the new Struts Action 2.0 which is scheduled for august 2006.

Changelog

For a complete list of all the changes, please refer to the [complete changelog](#)

Known Issues

[XSL Result](#) is known to have issues with Java 5.0.

OpenSymphony JIRA (113 issues)										
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
	WW-1265 DatePicker	Rene Gielen	Claus Ibsen			Closed	FIXED	Mar 22, 2006	Mar 23, 2006	
		locale/language problems								
	WW-1264 Fix broken calendar	Rene Gielen	Rene Gielen			Closed	FIXED	Mar 22, 2006	Mar 22, 2006	
		i18n_js scripts								
	WW-1263 config-broken - can not list interceptors correctly	Rainer Hermanns	Claus Ibsen			Closed	FIXED	Mar 22, 2006	Mar 22, 2006	
		interceptors correctly								
	WW-1262 showcase - some issues	Rene Gielen	Claus Ibsen			Closed	FIXED	Mar 22, 2006	Mar 22, 2006	
		issues								
	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	
		test directory								
		and a sample test case								
		to webapp "ant new" target								
	WW-1258 Ignoring ClientDiscardedException in the log file	Rene Gielen	Jeroen Vianen			Closed	WON'T FIX	Mar 20, 2006	Mar 21, 2006	
		ClientDiscardedException in the log file								
	WW-1256 Create a component to work along with ww:submit	Rene Gielen	Rene Gielen			Closed	FIXED	Mar 19, 2006	Mar 21, 2006	
		component to work along with ww:submit								
	WW-1254 zipped distribution should	Alexandru Popescu	Rainer Hermanns			Resolved	FIXED	Mar 19, 2006	Mar 19, 2006	
		zipped distribution								
		should								

	contain a project directory webwork-X.Y.Z						
WW-1253Default	Unassigned	Andres March	Resolved	FIXED	Mar 18, 2006	Mar 18, 2006	
	Action Support						
WW-1251Running	Claus Ibsen	Eric Molitor	Closed	FIXED	Mar 18, 2006	Mar 21, 2006	
	ant new in webapps fails						
WW-1249FileUpload	Dotm_jee	Brown	Resolved	FIXED	Mar 17, 2006	Mar 17, 2006	
	not using internationalized error message						
WW-1248DatePicker	Rene Gielen	Claus Ibsen	Closed	FIXED	Mar 17, 2006	Mar 17, 2006	
	single click not working						
WW-1247Document	tm_jee	Rene Gielen	Closed	FIXED	Mar 17, 2006	Mar 18, 2006	
	interceptor properties overriding on action configuration level						
WW-1246Document	tm_jee	Rene Gielen	Closed	FIXED	Mar 17, 2006	Mar 18, 2006	
	excludeMethods / includeMethods parameters for validator, token and workflow interceptor						
WW-1245Add an example for the Date component in showcase app	Rainer Hermanns	Rene Gielen	Closed	FIXED	Mar 16, 2006	Mar 20, 2006	
	Velocity doublesel tag example does not work as expected						
WW-1244Velocity	Rene Gielen	Rene Gielen	Closed	FIXED	Mar 16, 2006	Mar 18, 2006	
	doublesel tag example does not work as expected						

in						
showcase						
application						
WW-1243	doubleselection	Rene	Rene	Resolved	FIXED	Mar 16, 2006
	tag	Gielen	Gielen			Mar 18, 2006
	causes					
	exception,					
	indicating					
	invalid					
	stack					
	constitution					
	assumption					
WW-1242	action	Rainer	Ian	Closed	FIXED	Mar 16, 2006
	attribute	Hermanns	Roughley			Mar 17, 2006
	in form					
	ads					
	.action					
	onto					
	value					
WW-1241	Have	tm_jee	tm_jee	Resolved	FIXED	Mar 16, 2006
	ObjectFactoryDestroyable					Mar 16, 2006
	and					
	ObjectFactoryLifecycle					
	(in					
	addition					
	to the					
	current					
	ObjectFactoryInitializable)					
WW-1240	Improve	Rene	Rene	Closed	FIXED	Mar 15, 2006
	component	Gielen	Gielen			Mar 18, 2006
	parameter					
	documentation					
	to have					
	real					
	javadoc					
WW-1239	MultiPartRequest	Rainer	Nils-Hill	Resolved	FIXED	Mar 14, 2006
	uses	Hermanns				Mar 15, 2006
	Class.forName					
	resulting					
	in 500					
	error					
WW-1238	Fix	Rene	Rene	Resolved	FIXED	Mar 13, 2006
	Getting	Gielen	Gielen			Mar 22, 2006
	Started					
	Documentation					
WW-1237	Refine	Nils-Helge	Nils-Helge	Closed	FIXED	Mar 12, 2006
	logging	Garli	Garli			Mar 17, 2006
WW-1236	Missing	Unassigned	Nils-Helge	Resolved	FIXED	Mar 12, 2006
	test for		Garli			Dec 14, 2006
	null					
	'action'					
	property					
	in_start()					
	of URL					

component?					
WW-1235 JakartaM ClientRec Class	Ibsen	Ibsen	Closed	FIXED	Mar 11, 2006 Mar 11, 2006
- refactor to not used deprecated methods					
WW-1234 replace tm_jee tm_jee			Resolved	FIXED	Mar 11, 2006 Mar 11, 2006
deprecated way of geting Freemarker's Configuration in FreemarkerManager					
WW-1233 Add a tm_jee tm_jee			Resolved	FIXED	Mar 11, 2006 Mar 11, 2006
Plain Text Result					
WW-1231 ActionMap , Rainer yoshihara			Resolved	FIXED	Mar 10, 2006 Mar 16, 2006
no Hermann hidehiko support for multipart request					
WW-1230 Improve Nils-Helge Nils-Helge			Closed	FIXED	Mar 09, 2006 Mar 21, 2006
portal Garli Garli					
integration unit tests					
WW-1228 When Nils-Helge Jason			Closed	FIXED	Mar 07, 2006 Mar 17, 2006
deploying Garli Thorpe					
portlet that use the new webwork Jsr168Dispatcher class the portlet titles are unavailable in liferay 3.6.1					
WW-1227 Showcase Rainer Claus			Closed	FIXED	Mar 05, 2006 Mar 05, 2006
- token Hermann Ibsen examples					
WW-1226 Client Rainer Rainer			Closed	FIXED	Mar 04, 2006 Mar 18, 2006
side Hermann Hermanns					
JavaScript validation not working when using xwork-tiger					

WW-1225 <u>Clover</u> <u>report</u> <u>without</u> <u>log</u> <u>debug</u>	Rainer Claus Hermanns Ibsen	Closed	FIXED	Mar 03, 2006	Mar 04, 2006
WW-1224 <u>Portlet</u> <u>URL</u> <u>support</u> <u>doesn't</u> <u>follow</u> <u>JSR-168</u> <u>spec</u>	Nils-Helge Eric Garli Dalquist	Closed	FIXED	Mar 02, 2006	Feb 27, 2007
WW-1223 <u>div not</u> <u>working</u> <u>without</u> <u>delay > 0</u> <u>in ajax</u> <u>theme</u>	Ian Philip Roughley Luppens	Closed	FIXED	Mar 02, 2006	Mar 16, 2006
WW-1221 <u>Extending</u> <u>xhtml</u> <u>template</u>	Rainer Schava Hermanns Eugene	Closed	FIXED	Mar 01, 2006	Mar 17, 2006
WW-1220 <u>Some</u> <u>missing</u> <u>javadoc</u> <u>OpenSymphony</u> <u>copyright</u> <u>for</u> <u>recent</u> <u>submitted</u> <u>files</u>	Rene Claus Gielen Ibsen	Closed	FIXED	Mar 01, 2006	Mar 01, 2006
WW-1219 <u>Create a</u> <u>Button</u> <u>Tag</u>	Rene tm_jee Gielen	Resolved	FIXED	Feb 28, 2006	Mar 11, 2006
WW-1218 <u>Support</u> <u><input</u> <u>type=image</u> <u>...></u> <u>style</u> <u>html</u> <u>rendering</u> <u>for</u> <u><ww:submit</u> <u>/></u> <u>component</u>	Rene tm_jee Gielen	Resolved	FIXED	Feb 28, 2006	Mar 17, 2006
WW-1217 <u>ww:text</u> <u>tag</u> <u>returns</u> <u>an invalid</u> <u>value for</u> <u>an i18n</u> <u>key that</u> <u>starts</u> <u>with the</u> <u>same</u> <u>name as</u>	Rainer Nick Hill Hermanns	Closed	FIXED	Feb 27, 2006	Oct 03, 2006

an action variable	Rene Gielen	Rene Gielen	Resolved	CANNOT REPRODUCE	Feb 27, 2006	Mar 13, 2006
WW-1216Action specific I18n texts are not resolved for showcase CRUD example	Rainer Raible	Claus Ibsen	Closed	FIXED	Feb 26, 2006	Feb 26, 2006
WW-1215Validation - isDebugEnabled() missing	Rainer Raible	Claus Ibsen	Closed	FIXED	Feb 26, 2006	Feb 26, 2006
WW-1214ExecuteAndIntercept - new features, unit tests, improved javadoc and added examples to showcase	Rainer Raible	Claus Ibsen	Closed	FIXED	Feb 26, 2006	Feb 26, 2006
WW-1213ExecAndWith - logging spelling error	Alexandru Popescu	Claus Ibsen	Closed	FIXED	Feb 25, 2006	Feb 25, 2006
WW-1212Avoid excessive usage of ui attributes	tm_jee	tm_jee	Resolved	FIXED	Feb 25, 2006	Feb 27, 2006
WW-1211Add sample usage of ww:text tag to showcase example	Rene Gielen	Rene Gielen	Closed	IMPLEMENTED	Feb 24, 2006	Feb 25, 2006
WW-1210@ww.render render the same "label for" in multiple form	Patrick Lightbody	Aby Herwendo	Resolved	FIXED	Feb 24, 2006	Mar 05, 2006
WW-1209The validation	Patrick Lightbody	Matt Raible	Resolved	FIXED	Feb 24, 2006	Mar 05, 2006

WW-1208	FieldError	tm_jee	tm_jee	Resolved	FIXED	Feb 23, 2006	Feb 25, 2006
	<u>is not sent to the client as text/javascript</u>						
WW-1207	Add anchor attribute to the ww:url tag	Rainer Gielen	Philip Hermanns	Closed	FIXED	Feb 23, 2006	Feb 26, 2006
WW-1206	Request encoding mismatch	Rene Gielen	Tobias Järlund	Closed	FIXED	Feb 23, 2006	Feb 25, 2006
WW-1205	RegexField Validator cannot process white space string correctly	Robbin Gielen	Peter Järlund 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 26, 2006
WW-1202	Improve portal integration javadocs	Nils-Helge Garli	Nils-Helge Garli	Closed	FIXED	Feb 21, 2006	Mar 21, 2006
WW-1201	Remote form resultDiv not executing javascript	Ian Roughley	Steve Werner	Closed	FIXED	Feb 20, 2006	Mar 16, 2006
WW-1200	richtextedline is escaping content in ftl	Unassigned	Mike Porter	Resolved	FIXED	Feb 20, 2006	Feb 20, 2006
WW-1199	Submit tag	Patrick Lightbody	Patrick Lightbody	Resolved	FIXED	Feb 20, 2006	Feb 20, 2006

should support method and action attributes						
WW-1198Use <u>LinkedHashLightbody</u> instead of <u>TreeMap</u>	Patrick	Patrick Lightbody	Resolved	FIXED	Feb 20, 2006	Feb 20, 2006
WW-1196DoubleSelect <u>select does not allow a predefined value to be selected</u> (WW-1071)	Patrick	Alberto LightbodyOcampo	Resolved	FIXED	Feb 20, 2006	Mar 05, 2006
WW-1195labelposition <u>checkbox-tag does nothing</u>	Don Brown	Peter Westlin	Resolved	FIXED	Feb 20, 2006	Mar 23, 2006
WW-1194Checkbox <u>without a label generates a Frewamer template error</u>	Rene Gielen	Peter Westlin	Closed	FIXED	Feb 20, 2006	Feb 26, 2006
WW-1193bind.js <u>calls onLoad asynchronously</u>	Mike Porter	Mike Porter	Resolved	FIXED	Feb 18, 2006	Feb 18, 2006
WW-1192Generic <u>Support in xwork-tiger.jar does not work</u>	Rainer Hermanns	Corby Page	Closed	FIXED	Feb 18, 2006	Mar 21, 2006
WW-1188select-tag <u>with multiple="false" generates the same HTML as multiple="true"</u>	tm_jee	Peter Westlin	Resolved	FIXED	Feb 17, 2006	Feb 22, 2006
WW-1187Allow <u>easy overriding</u>	Rainer Hermanns	Matt Raible	Resolved	FIXED	Feb 16, 2006	Mar 17, 2006

of						
"excludeMethods"						
and						
"includeMethods"						
of						
ValidationInterceptor						
WW-1186	WW 2.2	Rainer Alex	Resolved	FIXED	Feb 16, 2006	Feb 20, 2006
logs		Hermann Shneyderman				
excetion						
if result						
of an						
action is						
null						
WW-1185	Add	Rainer Matt	Closed	FIXED	Feb 16, 2006	Aug 03, 2006
support		Hermann Raible				
for using						
Tiles with						
WebWork						
WW-1183	exclude	Alexandru Schava	Resolved	FIXED	Feb 15, 2006	Mar 17, 2006
methods		Popescu Eugene				
for token						
interceptor						
WW-1182@ww.token	tm_jee	Claus Ibsen	Resolved	FIXED	Feb 15, 2006	Feb 25, 2006
does not						
work						
with						
freemarker						
result						
WW-1181	Sitemesh	Alexandru Christian	Resolved	FIXED	Feb 15, 2006	Feb 25, 2006
freemarker		Popescu Meunier				
handled						
(FreemarkerPageFilter.java)						
should be						
locale						
aware						
WW-1177	add a	Rainer Mike	Closed	FIXED	Feb 14, 2006	Feb 26, 2006
debug		Hermann Porter				
attribute						
to						
ww:head						
tag						
WW-1176	Ajax	Alexandru Adam	Resolved	FIXED	Feb 14, 2006	Feb 02, 2007
elemnts:		Popescu Cuper				
remote						
form,						
remote						
div,...						
don't						
work on						
weblogic						
8.1						
WW-1175	HTTP	Ian Mike	Resolved	FIXED	Feb 14, 2006	Mar 15, 2006
Response		Roughley Porter				
Code						

WW-1174	SessionMap	Rainer	Dieter	Closed	FIXED	Feb 13, 2006	Feb 20, 2006
	<u>wrong for static convention served up via FilterDispatcher</u>						
WW-1173	Freemarker	tm_jee	Dennis	Closed	FIXED	Feb 12, 2006	Feb 18, 2006
	<u>characterset/locale configuration inconsistent with Webwork</u>						
WW-1172	submit	tm_jee	tm_jee	Resolved	FIXED	Feb 11, 2006	Feb 12, 2006
	<u>didn't include js event</u>						
WW-1170	Submit	Alexandru	Adam	Resolved	NOT A PROBLEM	Feb 10, 2006	Feb 16, 2006
	<u>button is Popescu Cuper always rendered in new line</u>						
WW-1169	migrate	Rainer	victorsosa	Closed	CANNOT REPRODUCE	Feb 09, 2006	Mar 15, 2006
	<u>to Hermanns FilterDispatcher</u>						
WW-1168	Template	Rene	Stanley	Resolved	CANNOT REPRODUCE	Feb 09, 2006	Mar 16, 2006
	<u>/ client Gielen Xu side validation related files (such as validate.js and style.css) references not resolvable in resulting html</u>						
WW-1167	form's client validation	Rene	Robbin	Closed	NOT A PROBLEM	Feb 09, 2006	Feb 20, 2006
	<u>should be executed after user</u>						

custom submit method							
WW-1165Missing setter for TLD declared attribute 'openTemplate' in TreeTag	Rene Gielen	Bhakta Koditipalli	Resolved	FIXED	Feb 07, 2006	Feb 08, 2006	
WW-1164Remove taglib definition in web.xml for the sample app	Nils-Helge Garli	Nils-Helge Garli	Closed	FIXED	Feb 07, 2006	Feb 12, 2006	
WW-1163Checkbox tag fails with freemarker error if no label set (for xhtml based themes)	Rene Gielen	Rene Gielen	Resolved	FIXED	Feb 07, 2006	Feb 07, 2006	
WW-1162Add support for detecting freemarker errors in UI tags tests	Rene Gielen	Rene Gielen	Resolved	FIXED	Feb 07, 2006	Feb 07, 2006	
WW-1159VelocityPageFilter needs to know about freemarker configuration	Rainer Brown	Rainer Hermanns	Resolved	FIXED	Feb 07, 2006	Mar 22, 2006	
WW-1158JasperReport View should support parameters for Report	Rainer Hermanns	Mariusz Smykula	Closed	NOT A PROBLEM	Feb 07, 2006	Feb 09, 2006	
WW-1157Re-name "action" parameter to avoid	Nils-Helge Garli	Nils-Helge Garli	Closed	IMPLEMENTED	Feb 07, 2006	Feb 07, 2006	

possible collission with other parameters						
WW-1156Velocity support	Nils-Helge Garli	Nils-Helge Garli	Closed	IMPLEMENTED	Feb 07, 2006	Feb 07, 2006
WW-1155Dispatcher only reads configuration values for encoding and locale during init - should read from configuration on every request	Rainer Raffler	Bruce Hermanns	Closed	FIXED	Feb 06, 2006	Feb 18, 2006
WW-1154Tag ww:text generate Exception related to ww:tree	Rainer Hermanns	Mariusz Smykula	Closed	DUPLICATED	Feb 06, 2006	Feb 06, 2006
WW-1153Combination of theme="simple" form components is breaking in certain instances	tm_jee	Ben Rometsch	Resolved	FIXED	Feb 06, 2006	Feb 12, 2006
WW-1152Theme information lost from parent form tag	tm_jee	Konrad Kamiński	Resolved	FIXED	Feb 06, 2006	Mar 03, 2006
WW-1151Remote Form submit displays result in new window (IE problem)	Ian Roughley	Adam Cuper	Resolved	FIXED	Feb 06, 2006	Feb 07, 2006
WW-1150When	Rene	Ben	Resolved	FIXED	Feb 06,	Feb 19,

setting a	Gielen	Rometsch			2006	2006
DatePicker						
form field						
as						
readonly="true"						
the user						
can still						
manipulate						
the field						
value						
WW-1149 Tree tag	Rainer	Gilles	Closed	FIXED	Feb 06, 2006	Feb 06, 2006
is has a	Hermanns	Dury's				
wrong						
name in						
taglib.tld						
WW-1148 QuickStart	Patrick	Stefan	Resolved	FIXED	Feb 05, 2006	Mar 05, 2006
does not	Lightbody	Arentz				
work on						
OS X;						
throws						
exception						
WW-1147 Add	tm_jee	tm_jee	Resolved	FIXED	Feb 05, 2006	Feb 20, 2006
tooltip						
support						
to						
components						
WW-1146 Add a	tm_jee	tm_jee	Resolved	FIXED	Feb 05, 2006	Feb 20, 2006
simple						
RichTextEditor						
component						
to						
webwork						
WW-1145 JSTL	Rainer	Mathias	Resolved	FIXED	Feb 04, 2006	Mar 17, 2006
support	Hermanns	Bogaert				
not						
SiteMesh						
compatible						
WW-1144 redirect-at	Rene	Eric	Closed	FIXED	Feb 03, 2006	Nov 04, 2006
result	Gielen	Molitor				
cannot						
be						
configured						
with GET						
parameters						
in						
xwork.xml						
WW-1135 When	Rainer	Eric	Closed	FIXED	Jan 31, 2006	Mar 21, 2006
using the	Hermanns	Molitor				
action						
tag and						
executeResults						
in a						
freemarker						
template						

on a jsp page the JSPWriter buffer isn't being flushed	WW-1131	Request	Andres March	Nicholaus Shupe	Closed	NOT A PROBLEM	Jan 28, 2006	Feb 17, 2006
WW-1011 Problem with SiteMesh tags and ww:action	WW-1011	Problem	Patrick Lightbody	Patrick Lightbody	Closed	FIXED	Dec 13, 2005	Mar 18, 2006
WW-1003 ActionCom injection and xwork.xml overlap	WW-1003	ActionCom	Patrick Lightbody	Patrick Lightbody	Resolved	FIXED	Dec 11, 2005	Mar 21, 2006
WW-978 ww:sort and ww:generators tags are broken	WW-978	ww:sort	Rainer Hermanns	Janari Pöld	Closed	FIXED	Nov 30, 2005	Feb 16, 2007
WW-959 Access to meta properties in freemarker decorators (without sitemesh tags).	WW-959	Access to meta properties	Rainer Hermanns	Kristoffer Skjutare	Closed	FIXED	Nov 16, 2005	Mar 16, 2006
WW-924 Internationalization document complete Spring integration	WW-924	Internationalization	Patrick Gielen	Patrick Lightbody	Closed	FIXED	Oct 26, 2005	Mar 21, 2006
WW-852 Document Setting the character encoding on the request causes all form parameters to be erased on Oracle	WW-852	Document	Rainer Hermanns	Patrick Lightbody	Resolved	FIXED	Oct 07, 2005	Mar 15, 2006
WW-822 Setting the character encoding on the request causes all form parameters to be erased on Oracle	WW-822	Setting	Rene Gielen	accbank	Closed	CANNOT REPRODUCE	Sep 03, 2005	Mar 13, 2006

	<u>9j</u>						
WW-808	ParamTag	Claus Ibsen	John Patterson	Closed	FIXED	Jul 15, 2005	Mar 17, 2006
	<u>does not convert to string</u>						
WW-805	ww:date	Rainer tag	Patrick Hermann Lightbody	Closed	FIXED	Jul 11, 2005	Mar 21, 2006
WW-724	FileUpload	Rainer Error	toshihito Hermannsogami	Resolved	CANNOT REPRODUCE	Jan 18, 2005	Feb 17, 2006
	<u>File Type</u>						
	<u>Tiff , PDF</u>						
	<u>] or [</u>						
	<u>network</u>						
	<u>Drive File</u>						
	<u>1</u>						
WW-649	Field errors	tm_jee	Matt Raible	Resolved	FIXED	Sep 28, 2004	Mar 02, 2006
	<u>should be displayed</u>						
	<u>in the order</u>						
	<u>they're in the POJO</u>						

WebWork 2.2.3

This page last changed on Aug 22, 2006 by [phil](#).

WebWork 2.2.3 Release Notes

Key Changes

UI and Views

- New and improved components: [datepicker](#), [head](#), [form](#), [radio](#), [tree](#), [text](#)
- Various bug fixes for the [ajax theme](#)
- Improved xslt and stream result

Tools

- [Quickstart](#) Support for arbitrary webapp deployment
- The new [debug](#) interceptor provides an ajax console to test your ognl expressions

Misc

- Better support for i18n and l10n
- Support for Java 5 enumeration type conversion
- Encodings during multipart requests are now properly handled
- Better and more transparent error handling

Migration Notes

The WebWork 2.2.3 release was aimed at fixing bugs for the current 2.2.x users. We will most likely not do another WW release, since all the developer attention now goes to the Struts 2.0 and XWork 2.0 projects, which should hit beta in the next few months. Users are advised to check their redirect and redirect-action results, as well as the various ajax related elements such as tabbed panes and datepickers. We'd like to thank everyone who helped us out on this release, and hope to see the same level of commitment and dedication over at Struts 2.0.

Changelog

For a complete list of all the changes, please refer to the [complete changelog](#)

OpenSymphony JIRA (80 issues)										
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
	WW-1322	ww:head tag includes parameters in URLs to stylesheets	Rainer	Patrick Hermann		Resolved	FIXED	Jul 13, 2006	Feb 06, 2007	
	WW-1323	calendar tag produces URLs with included parameters	Rainer	Patrick Hermann		Resolved	FIXED	Jul 13, 2006	Jul 13, 2006	
	WW-1331	The Jsr168Dispatcher does not initialize the DispatcherUtils	Nils-Helge	Nils-Helge Garli		Closed	FIXED	Jul 30, 2006	Aug 14, 2006	
	WW-1339	Redirect Action Result does no longer support paramterized values	Rainer	Rainer Hermann		Closed	CANNOT REPRODUCE	Aug 14, 2006	Aug 22, 2006	
	WW-492	XSLT Views + XML Mapping + HashMap support	Rainer	kiran Hermann		Closed	FIXED	Mar 08, 2004	Jun 26, 2006	
	WW-1319	Improve QuickStart so arbitrary webapps can be deployed as well	Patrick	Bill Lynch		Resolved	FIXED	Jul 11, 2006	Jul 17, 2006	
	WW-1318	Deploying two webwork portlets in Jetspeed 2.0 throws ClassCastException	Nils-Helge	Gunnar Rohde		Resolved	WON'T FIX	Jul 11, 2006	Aug 09, 2006	

WW-1284	radiomap	Rainer	Eric	Closed	FIXED	May 30, 2006	May 31, 2006
	<u>does not honour Boolean keys.</u>	Hermanns	Molitor				
WW-1296	remote	tm_jee	tm_jee	Closed	FIXED	Jun 10, 2006	Jun 18, 2006
	<u>div tag should be able to not auto start</u>						
WW-1285	optiontransfer	Rainer	Eric	Closed	FIXED	May 30, 2006	May 31, 2006
	<u>tag does not honour doubleCssClass</u>	Molitor					
WW-1300	Add example using tree component	tm_jee	tm_jee	Closed	FIXED	Jun 16, 2006	Jun 18, 2006
WW-1298	Tree tag - setting some attribute have no effect.	tm_jee	tm_jee	Resolved	FIXED	Jun 14, 2006	Jun 14, 2006
	<u>- setting some attribute have no effect.</u>						
WW-1085	Datepicker i18n related javascript errors	Rainer	Patrick	Resolved	FIXED	Jan 13, 2006	Jun 18, 2006
	<u>i18n related javascript errors</u>	Hermanns	Lightbody				
WW-1283	form tag doesn't honour actionAlias!method	Rainer	Eric	Closed	FIXED	May 30, 2006	May 31, 2006
	<u>doesn't honour actionAlias!method</u>	Hermanns	Molitor				
WW-1294	empty.ftl should be in simple theme instead of xhtml theme	tm_jee	tm_jee	Closed	FIXED	Jun 10, 2006	Jun 18, 2006
	<u>should be in simple theme instead of xhtml theme</u>						
WW-1097	AJAX tags not working with IE 5.5	Rainer	Rainer	Closed	WON'T FIX	Jan 17, 2006	Jun 18, 2006
	<u>tags not working with IE 5.5</u>	Hermanns	Hermanns				
WW-1282	Form Tag needs accept-charset attribute	Rainer	Eric	Closed	FIXED	May 30, 2006	May 31, 2006
	<u>needs accept-charset attribute</u>	Hermanns	Molitor				
WW-1270	tabbed pane	tm_jee	Kevin Werner	Resolved	CANNOT REPRODUCE	Mar 25, 2006	Jun 21, 2006

shows 2 panels in some circumstances	Rainer Hermanns	tm_jee	Closed	FIXED	Mar 25, 2006	Jun 18, 2006
WW-1268Text component should override useBody() to return true	Rainer Hermanns	tm_jee	Closed	FIXED	Sep 26, 2004	Jun 18, 2006
WW-646 HTML Editor Component	Ricardo Lecheta	tm_jee	Closed	FIXED	Sep 26, 2004	Jun 18, 2006
WW-1260Quickstart does not work well with DWR	Francisco Assis Rosa	Rainer Hermanns	Closed	NOT A PROBLEM	Mar 21, 2006	Jun 18, 2006
WW-1267Encoding problem with URL tag when includeParams attribute is GET	Rainer Hermanns	tm_jee	Resolved	FIXED	Mar 25, 2006	Jun 20, 2006
WW-1286Freemarker Mapeget cannot be created through Spring	tm_jee	Rainer Hermanns	Closed	FIXED	Jun 05, 2006	Feb 19, 2007
WW-1259<ww:head> tag in ajax theme causes secure/non-secure content prompt in IE	Jason Jones	Rainer Hermanns	Resolved	WON'T FIX	Mar 21, 2006	Jun 23, 2006
WW-1269ActionAction required to have a methodName attribute as well.	Rainer Hermanns	tm_jee	Closed	FIXED	Mar 25, 2006	Jun 18, 2006
WW-1311Unnecessary dependency on portlet API	Logi Ragnarsson	Rainer Hermanns	Closed	NOT A PROBLEM	Jun 27, 2006	Jun 27, 2006
WW-1278Prefix failed	Rainer Hermanns	tm_jee	Closed	FIXED	Mar 28, 2006	Jun 18, 2006

when						
Multipart						
form is						
submitted						
WW-1018validation	Jason	jacky hua	Closed	WON'T FIX	Dec 18, 2005	Jun 18, 2006
problem	Carreira					
for						
webwork's						
!command						
pattern						
action						
WW-827 Error in a	Ian	Ian	Closed	FIXED	Sep 15, 2005	Jun 18, 2006
tags	Roughley	Roughley				
theme						
template						
causes						
fallback						
WW-1304TLD for	Rainer	Rainer	Closed	FIXED	Jun 22, 2006	Jun 26, 2006
FormTag	Hermann	Hermanns				
is						
missing						
accept						
charset						
(error in						
Form						
component						
xdoclet						
tags)						
WW-647 Client-side	Patrick	Matt	Resolved	WON'T FIX	Sep 28, 2004	Jun 23, 2006
validation	Lightbody	Raible				
show all						
validation						
errors						
WW-1302Uploaded	Rainer	Gilles	Closed	FIXED	Jun 20, 2006	Jun 26, 2006
filenames	Hermann	Dury				
are						
converted						
to						
lowercase						
WW-1317ajax	tm_jee	tm_jee	Resolved	FIXED	Jul 04, 2006	Jul 09, 2006
capabilities						
with						
css_xhtml						
theme						
WW-1307Datepicker	tm_jee	Eric	Closed	FIXED	Jun 25, 2006	Jun 26, 2006
doesn't		Molitor				
work						
with						
swedish						
locale						
(with						
workarounds)						
WW-1293ComponentTag	tm_jee	tm_jee	Closed	FIXED	Jun 10, 2006	Jun 26, 2006
(GenericUIBean						

component)						
should						
allow						
templateDir						
to be set						
as						
attribute						
as well						
WW-1315Ajax	tm_jee	dusty pearce	Resolved FIXED	Jun 29, 2006	Jul 01, 2006	
form-end						
template						
missing						
tooltip.js						
WW-1316UITag	tm_jee	tm_jee	Resolved FIXED	Jul 04, 2006	Jul 04, 2006	
example						
in WW is						
not						
working						
WW-1279Interceptor	Rainer param	Eric Hermanns Molitor	Closed FIXED	May 30, 2006	Jun 26, 2006	
includeMethods						
doesnt						
work						
with JS						
validation						
WW-1305ClassCastException	Rainer in	Eric Hermanns Molitor	Closed FIXED	Jun 25, 2006	Jun 26, 2006	
LocalizedTextUtils						
in Portlet						
when						
doing a						
submit to						
a model						
driven						
action						
WW-1306Result	Rainer Url not	Eric Hermanns Molitor	Closed DUPLICATE	Jun 25, 2006	Jul 01, 2006	
encoded						
WW-1299Demo	Nils-Helge Portlet	Gunnar Garli Rohde	Resolved NOT A PROBLEM	Jun 15, 2006	Jul 04, 2006	
under						
Jetspeed						
throws						
ognlException						
WW-1291ActionContextCleanUp	Rainer can fail in	Pete Hermanns Matern	Closed FIXED	Jun 07, 2006	Jun 26, 2006	
WebLogic						
8.1						
WW-1325Tabpanel	tm_jee	tm_jee	Resolved FIXED	Jul 16, 2006	Jul 16, 2006	
panel						
does						
display						
its						
content						

WW-1308	Java 5	Rainer Hermanns	Tamara Cattivelli	Closed	FIXED	Jun 26, 2006	Jun 27, 2006
WW-1324	Debug tag does not	Rainer Hermanns	Rainer Hermanns	Closed	FIXED	Jul 13, 2006	Dec 18, 2006
WW-1289	Quickstart does not load classes from WEB-INF/classes first	Patrick Lightbody	Nick Hill	Resolved	FIXED	Jun 06, 2006	Jul 17, 2006
WW-945	Global Results conflict	tm_jee	Onyeje Bose	Resolved	FIXED	Nov 08, 2005	Jul 13, 2006
WW-1330	<ww:action does not pass <ww:param> values to action	tm_jee	Pat Niemeyer	Resolved	NOT A PROBLEM	Jul 29, 2006	Aug 09, 2006
WW-1326	SessionMap needs an invalidate method	Patrick Lightbody	Patrick Lightbody	Resolved	FIXED	Jul 18, 2006	Jul 18, 2006
WW-1337	utf-8 and commons upload	Rainer Hermanns	Rainer Hermanns	Closed	DUPLICATE	Aug 11, 2006	Aug 14, 2006
WW-1191	JavaScript validation (non-Ajax) doesn't look for an existing message before adding a new validation error	tm_jee	Matt Raible	Resolved	CANNOT REPRODUCE	Feb 17, 2006	Jul 30, 2006
WW-1332	Exception thrown when	tm_jee	Francisco Assis Rosa	Resolved	FIXED	Jul 30, 2006	Aug 02, 2006

contentLength						
expression						
supplied						
in stream						
result						
xwork.xml						
definition						
WW-1303 notify appears to be freemarker reserved word	Rainer Hermanns	Cris Daniluk	Closed	CANNOT REPRODUCE	Jun 20, 2006	Aug 14, 2006
WW-1290 Debuggability for WebWork 2.2.2	Rainer Hermanns	Tamara Cattivelli	Closed	FIXED	Jun 07, 2006	Aug 10, 2006
WW-1336 Freemarker result on WebLogic causes problems	Patrick Lightbody	Patrick Lightbody	Resolved	FIXED	Aug 09, 2006	Aug 09, 2006
WW-1272 improve "ant new" for creating webapp to allow directory webapp to be created under	tm_jee	tm_jee	Resolved	WON'T FIX	Mar 26, 2006	Aug 21, 2006
WW-1335 commons Upload encoding bug	tm_jee	tm_jee	Resolved	FIXED	Aug 06, 2006	Aug 18, 2006
WW-1328 Add If, Else and ElseIf to freemarker model	tm_jee	tm_jee	Resolved	FIXED	Jul 22, 2006	Aug 29, 2006
WW-769 UITags do not evaluate id attribute	tm_jee	John Patterson	Closed	FIXED	Apr 12, 2005	Aug 22, 2006
WW-1288 Strange behavior with <ww:action> tag	tm_jee	Nick Hill	Resolved	FIXED	Jun 06, 2006	Mar 20, 2007
WW-1266 URL tag - includePath	Rainer Hermanns	Claudio Miranda	Closed	FIXED	Mar 25, 2006	Jun 18, 2006

WW-1280	Unwanted <u>locale-for-Herma</u> ns	Rainer Eric Molitor of checkbox values	Closed	FIXED	May 30, 2006	Jun 18, 2006
WW-1255	XSLTResu l - 3 issues	Rainer Claus Hermanns Ibsen	Closed	FIXED	Mar 19, 2006	Jun 26, 2006
WW-1292	theme.properties	Cris not read from filesystem	Resolved	FIXED	Jun 08, 2006	Dec 11, 2006
WW-1313	WebWork form tags do not have an accesskey attribute	tm_jee Jennifer Grucza	Resolved	FIXED	Jun 27, 2006	Jul 02, 2006
WW-1287	Java 5 <u>Enumeration</u> Support	Rainer Tamara Hermanns Cattivelli	Closed	FIXED	Jun 06, 2006	Jun 26, 2006
WW-1320	Invalid pointer	Rainer Nils-Helge Hermanns Garli to the lib/freemarker folder in the build-portlet target in the build.xml for the webapps.	Closed	FIXED	Jul 12, 2006	Jul 13, 2006
WW-1312	ww:select tag does not support optgroup	tm_jee Jennifer Grucza	Resolved	FIXED	Jun 27, 2006	Jul 07, 2006
WW-1295	Tabbed Panel: All	Rainer Richard Hermanns Lawrence tabs seen when first tab has remote="true"	Closed	FIXED	Jun 10, 2006	Jul 16, 2006
WW-1334	submit tag with image	Rainer justin Hermanns fields	Closed	FIXED	Aug 01, 2006	Aug 10, 2006

WW-1329	redirect-action	tm_jee	Jennifer Grucza	Resolved	FIXED	Jul 24, 2006	Jul 31, 2006
	type does not forward action correctly						
WW-1297	radiomap	Rainer Lang	Hermanns	Closed	FIXED	Jun 11, 2006	Dec 18, 2006
	formats long numbers with commas						
WW-1327	ww:param	Rainer Hermanns	Sean Moriarty	Resolved	NOT A PROBLEM	Jul 19, 2006	Aug 10, 2006
	not functioning when used with ww:include						
WW-1274	Wrong end	Alexandru Popescu	Parker Wang	Closed	WON'T FIX	Mar 27, 2006	Aug 10, 2006
	index used when pick path info from the request URI in RequestUtils.getServletPath(request).						
WW-1310	readonly attribute	Rainer Hermanns	Dieter van Baarle	Closed	FIXED	Jun 26, 2006	Feb 27, 2007
	on ww:textfield does not behave as a boolean						
WW-978	ww:sort ww:generate	Rainer Hermanns	Janari Pöld	Closed	FIXED	Nov 30, 2005	Feb 16, 2007
	and ww:subset tags are broken						
WW-980	how to reload xwork configuration	tm_jee	victorsosa	Resolved	FIXED	Dec 03, 2005	Jun 23, 2006
WW-770	Validation	Unassigned	Alex Shneyderman	Closed	WON'T FIX	Apr 15, 2005	Jun 18, 2006
	should be aware of						

namespace					
that					
action					
lives in					
WW-1271RequestMap	Rainer Matt		Closed	FIXED	Mar 25, Jun 18,
	not Hermanns Crawford				2006 2006
clearing					
entries					
when					
RequestMap.entries					
is called.					
WW-1338URI is	Philip Binil		Closed	FIXED	Aug 14, Aug 14,
	/webwork Lippens Thomas				2006 2006
not					
"webwork"					
in 2.2.2					

WebWork 2.2.4

This page last changed on Apr 12, 2007 by [rainierh](#).

WebWork 2.2.4 Release Notes

Migration Notes

The WebWork 2.2.4 release was aimed at fixing bugs for the current 2.2.x users. Users are advised to check their redirect and redirect-action results, as well as the various ajax related elements such as tabbed panes and datepickers. We'd like to thank everyone who helped us out on this release, and hope to see the same level of commitment and dedication over at Struts 2.0.

Changelog

For a complete list of all the changes, please refer to the [complete changelog](#)

OpenSymphony JIRA (14 issues)											
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due	
	WW-1343	Else tag body displayed when it shouldn't	tm_jee	Gilles Dury		Resolved	FIXED	Aug 29, 2006	Aug 30, 2006		
	WW-1345	ListTags do not allow evaluation of listValue from ValueStack	Rainer	Rainer Hermanns		Closed	FIXED	Aug 29, 2006	Feb 06, 2007		
	WW-1366	FileUpload interceptor only calls acceptFile on first file	Jasper Rosenberg			Resolved	FIXED	Oct 12, 2006	Feb 06, 2007		
	WW-1346	URL Tag includePath default attribute value should be configurable	Rainer	Rainer Hermanns		Closed	FIXED	Aug 29, 2006	Feb 19, 2007		
	WW-1340	ScopeInterceptor caches	Rainier Warren Hermanns	Blanchet		Resolved	FIXED	Aug 16, 2006	Aug 29, 2006		

key in instance variable	Rainer	Rainer	Closed	FIXED	Aug 30, 2006	Aug 30, 2006
WW-1348 Option Tra... Refer Selector Rainer attribute Hermanns	Hermanns	Hermanns				
doubleDisabled does not work						
WW-1341 JakartaM... Refer Selector Rainer mismatch Hermanns	on		Closed	NOT A PROBLEM	Aug 23, 2006	Aug 29, 2006
with Brandis commons-fileupload library						
WW-1350 ant dist target Hermanns	Rainer	Rainer	Closed	FIXED	Aug 31, 2006	Sep 02, 2006
does not package ivyconf.xml	Hermanns	Hermanns				
WW-1344 Bug in WebWork TLD	Rainer	James	Resolved	FIXED	Aug 29, 2006	Aug 29, 2006
Hermanns House TLD	Hermanns	House				
WW-1349 URL Tag cause duplicates query string	tm_jee	tm_jee	Resolved	FIXED	Aug 31, 2006	Aug 31, 2006
WW-1347 Bug in doubleselect (same as WW-1297)	Rainer	Martin	Closed	FIXED	Aug 30, 2006	Feb 19, 2007
Hermanns Berg	Hermanns	Berg				
WW-1381 invalid html 4.0 generated by ww:head tag	Rainer	Urban	Closed	NOT A PROBLEM	Nov 23, 2006	Nov 23, 2006
Hermanns Lindberg	Hermanns	Lindberg				
WW-1342 Freemarket links are broke on Webwork WIKI	Rainer	Troy	Closed	FIXED	Aug 25, 2006	Aug 29, 2006
Hermanns Kelley	Hermanns	Kelley				
WW-978 ww:sort ww:generate and ww:subset tags are broken	Rainer	Janari	Closed	FIXED	Nov 30, 2005	Feb 16, 2007
Hermanns Pöld	Hermanns	Pöld				

WebWork 2.2.5

This page last changed on Apr 12, 2007 by [phil](#).

WebWork 2.2.5 Release Notes

Key Changes

Probably the most important bug fix is related to the XML parsing of xwork configuration files, where some XML parsers treated whitespace different than others, resulting in elements being read incorrectly. Apart from that, we fixed various component (mostly Ajax) related bugs, improved [Freemarker](#) support, upgraded to [Dojo](#) 0.4 and generally fine-tuned the framework.

See below for the complete list of changes.

Migration Notes

This 2.2.5 release, 6 months after [WebWork 2.2.4](#), marks the final chapter for the [WebWork](#) framework. [Struts 2](#), its successor, had its first GA release, and in a final push, the [WebWork](#) team wanted to do a big effort to bring in as many optimizations and bug fixes as possible before shifting developer focus.

This release is fully backwards compatible with [WebWork 2.2.4](#). You can just replace the existing [WebWork](#) and XWork jars with the news ones from 2.2.5.

Like always, we would like to thank both regular users and contributors, for helping out, submitting patches, testing, and providing general feedback. The development of [WebWork](#) 2 and XWork 1 slowly grinds to a halt, so if you're looking for bleeding edge technologies or new features, you are more than welcome to try [Struts 2](#).

Changelog

For a complete list of all the changes, please refer to the [complete changelog](#)

OpenSymphony JIRA (93 issues)										
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
	WW-1273 freemarker	Philip Luppens	Vladimir Olenin			Resolved	NOT A PROBLEM	Mar 27, 2006	Feb 10, 2007	
	WW-1408 Race condition in	Unassigned	Philip Luppens			Resolved	FIXED	Jan 02, 2007	Jan 08, 2007	

WW-1372	ww:div	tm_jee	Konstantin Pribluda	Resolved	FIXED	Oct 27, 2006	Jan 10, 2007
	shall be block directive in velocity view						
WW-1416	Ajax	tm_jee	tm_jee	Resolved	FIXED	Jan 10, 2007	Jan 14, 2007
	validation broken when labelposition is "top"						
WW-1371	Force	tm_jee	Konstantin Pribluda	Resolved	FIXED	Oct 24, 2006	Jan 10, 2007
	URL bean to return itself in setters instead of void						
WW-1370	PicoObjectFactory	Tom	Konstantin Schneider Pribluda	Resolved	FIXED	Oct 20, 2006	Feb 06, 2007
	produces singleton instances of validators & other stuff						
WW-1413	Numerical	Unassigned	Edmorthy String as KeyProperty	Resolved	NOT A PROBLEM	Jan 08, 2007	Feb 04, 2007
	in Collection Type Conversion causes NPE (and sometimes ognl Exception).	Wonil Lee					
WW-711	combobox	Philip	Quake	Closed	WON'T FIX	Dec 21, 2004	Feb 10, 2007
	tag does not support listKey and listValue	Luppens	Wang				
WW-796	SetTag	Philip	Benz	Resolved	FIXED	Jun 15, 2005	Feb 06, 2007
	"scope" attribute do not put the object to	Luppens	Shen				

WW-1407	Support	tm_jee	Radhakrishnan J	Resolved	WON'T FIX	Dec 26, 2006	Feb 06, 2007
	stack if the scope is given.						
WW-1399	upgrade	tm_jee	tm_jee	Resolved	FIXED	Dec 16, 2006	Jan 03, 2007
	to dojo 0.4						
WW-1401	Allow Ognl to be swapable with other expression language	Unassigned	tm_jee	Resolved	WON'T FIX	Dec 19, 2006	Jan 08, 2007
WW-1369	Java 5 Enum values are not handled properly by Radio jsp tag	tm_jee	Vlad Kravchenko	Resolved	FIXED	Oct 20, 2006	Nov 07, 2006
WW-1361	ww:double Robert J. javascript Hermann Longman uses name instead of id, resulting in array names that don't work	tm_jee	tm_jee	Resolved	FIXED	Sep 27, 2006	Feb 07, 2007
WW-1400	URL tag includeParams get functionality does not work under websphere	tm_jee	tm_jee	Resolved	FIXED	Dec 17, 2006	Dec 17, 2006
WW-600	Client-side validation Philip Luppens Matt Raible doesn't work as	Philip Luppens	Matt Raible	Closed	WON'T FIX	Jul 21, 2004	Feb 10, 2007

advertis						
WW-1418optgroup.ftl	tm_jee	tm_jee	Resolved	FIXED	Jan 23, 2007	Jan 23, 2007
formats						
long						
numbers						
with						
commas						
WW-1229Make	Philip	tm_jee	Resolved	FIXED	Mar 08, 2006	Feb 23, 2007
WebWork's	Luppens					
Cookbook						
uptodate						
WW-1403URL tag	tm_jee	tm_jee	Resolved	FIXED	Dec 20, 2006	Dec 20, 2006
should						
support						
not						
escaping						
ampersand						
WW-1390Adding	tm_jee	Mark	Resolved	FIXED	Dec 04, 2006	Dec 23, 2006
hasKey		Chaimungkalanont				
to the						
TextProvider						
interface						
WW-1409Cookie	tm_jee	Philip	Resolved	FIXED	Jan 02, 2007	Jan 09, 2007
interceptor		Luppens				
WW-1405HTML	tm_jee	Yoav	Resolved	FIXED	Dec 21, 2006	Dec 30, 2006
<script>		Shapira				
tags in						
templates						
missing						
type						
attribute,						
causing						
invalid						
HTML						
WW-1380Validator	tm_jee	tm_jee	Resolved	FIXED	Nov 20, 2006	Nov 25, 2006
defined						
in xml						
should be						
able to						
parse						
xml tag						
values						
against						
value						
stack						
WW-1411OptionTransferSelect	tm_jee	tm_jee	Resolved	FIXED	Jan 02, 2007	Jan 02, 2007
doubleSize						
is						
ignored						
in lieu of						
size						
property						
WW-1383Missed	tm_jee	luckystrike	Resolved	FIXED	Nov 28, 2006	Jan 10, 2007
<@ww.treenode..>						

WW-1412			ParamRemoval	in freemarker tags	Resolved	FIXED	Jan 04, 2007	Jan 06, 2007
WW-1384			tm_jee	tm_jee broken with handling static resources under WebSphere 6	Resolved	FIXED	Nov 28, 2006	Nov 28, 2006
WW-1415			Philip Luppens	James David Issues with Ajax Validation	Resolved	CANNOT REPRODUCE	Jan 09, 2007	Feb 10, 2007
WW-1368			Philip Luppens	Li Shaowei bug with ajax form	Resolved	FIXED	Oct 19, 2006	Feb 10, 2007
WW-1425			tm_jee	tm_jee Sitemesh's applyDecorator tag using Freemarker Transform	Resolved	FIXED	Feb 15, 2007	Feb 16, 2007
WW-1333			Philip Luppens	Per Rolfhamre with nested panels in an iterator	Resolved	CANNOT REPRODUCE	Jul 31, 2006	Feb 10, 2007
WW-1402			Unassigned	Sibhel Katchi entry in xwork.xml not being read correctly.	Resolved	FIXED	Dec 19, 2006	Feb 04, 2007
WW-1363			Rainer PortletSessionManager	Thomas Roka-Aardal not serializable	Resolved	FIXED	Oct 02, 2006	Jan 08, 2007
WW-1029			Unassigned	Dhruva Reddy support for multiple URL extensions	Resolved	WON'T FIX	Dec 21, 2005	Feb 10, 2007
WW-1388			tm_jee	tm_jee ParameterFilterInterceptor to webwork-default.xml	Resolved	FIXED	Dec 03, 2006	Dec 03, 2006
WW-1420			tm_jee	tm_jee doesn't	Resolved	FIXED	Jan 25, 2007	Feb 02, 2007

read					
theme.properties					
when it is					
in					
webapp					
path					
WW-1382Have an interceptor that invalidates http session	tm_jee	tm_jee	Resolved FIXED	Nov 23, 2006	Jan 20, 2007
WW-1387Flash result type	tm_jee	tm_jee	Resolved FIXED	Dec 03, 2006	Dec 11, 2006
WW-1406template/tm_jee/radiowiad.ftl cann't checked the radio when the name is Boolean	tm_jee	tm_jee	Resolved NOT A PROBLEM	Dec 23, 2006	Feb 02, 2007
WW-1396cssStyle and cssClass attribute of datepicker tag doesn't style the image	tm_jee	tm_jee	Resolved FIXED	Dec 06, 2006	Dec 11, 2006
WW-1393Dispatcher threadlocal cleanup does not clean up the threadlocal	Tom Schneider	tm_jee	Resolved NOT A PROBLEM	Dec 06, 2006	Feb 02, 2007
WW-1352duplicate tabindex in checkbox.ftl	tm_jee	tm_jee	Resolved FIXED	Sep 07, 2006	Sep 07, 2006
WW-1395Jsr168Dispatcher does not destroy the Dispatcher properly	tm_jee	tm_jee	Resolved FIXED	Dec 06, 2006	Dec 21, 2006
WW-1394RequestContext tm_jee is not cleaned up on undeploy	tm_jee	tm_jee	Resolved NOT A PROBLEM	Dec 06, 2006	Jan 08, 2007

WW-1355	SessionAttachment	Paul Boudreaux	Resolved	FIXED	Sep 14, 2006	Sep 16, 2006
	<u>Session</u> <u>map</u> <u>clear()</u> <u>method</u> <u>not</u> <u>functioning</u> <u>correctly.</u>					
WW-1414	Tree	tm_jee tm_jee component	Resolved	FIXED	Jan 09, 2007	Jan 09, 2007
	<u>not</u> <u>working</u> <u>as</u> <u>expected</u> <u>due to</u> <u>upgrade</u> <u>of dojo</u> <u>from</u> <u>0.2.x to</u> <u>0.4.1</u>					
WW-1397	XSLTResult	tm_jee tm_jee throws NullPointerException	Resolved	FIXED	Dec 06, 2006	Dec 06, 2006
WW-1353	FreeMarker	Rainer Marc template Hermann Lustig error! error.ftl]	Closed	FIXED	Sep 11, 2006	Feb 08, 2007
WW-1356	Select	tm_jee tm_jee Component does not preselect header	Resolved	FIXED	Sep 19, 2006	Sep 19, 2006
WW-1398	error.ftl	tm_jee tm_jee throws exception due to part of the model it uses is not available	Resolved	FIXED	Dec 11, 2006	Dec 11, 2006
WW-291	Too sophisticated	Tom Schneider Mike Mosiewicz exception handling in servlet dispatcher	Resolved	FIXED	Sep 04, 2003	Feb 02, 2007
WW-1391	NULPointerException	Vlad in Schneider Kravchenko UrlHelper.buildUrl method.	Resolved	FIXED	Dec 05, 2006	Feb 05, 2007
WW-1386	ScopeIntrospection	Daniel Uribe assigned Daniel Uribe handling	Closed	FIXED	Nov 29, 2006	Feb 02, 2007

of null values for session attributes breaks under some circumstances	WW-1122	Submit component	Rene Gielen	Onyeje Bose	Resolved	FIXED	Jan 25, 2006	Mar 02, 2007
sets the field value rather than a boolean on the stack.	WW-1360	UIBean component	Tom Schneider	Joseph Pemberton	Resolved	FIXED	Sep 23, 2006	Mar 02, 2007
sets malformed default ID with ONGL expression for name parameter	WW-1379	FileUpload component	Tom intercept and localized messages problem	Philip SchneiderLuppens	Resolved	WON'T FIX	Nov 20, 2006	Feb 18, 2007
and URLHelper class do not handle multiple request parameters with the same name correctly when includeParams="get"	WW-1376	URL component	Tom Schneider	Daniel Uribe	Resolved	FIXED	Nov 13, 2006	Feb 03, 2007
tm_jee validation (using dwr)	WW-1427	ajax	tm_jee validation (using dwr)	tm_jee	Resolved	FIXED	Mar 10, 2007	Mar 10, 2007

generates					
js_error					
on firefox					
when					
there's a					
field not					
defined					
in					
*-validation.xml					
WW-1404	Token	Tom Martin	Resolved FIXED	Dec 21, 2006	Mar 03, 2007
	session	SchneiderGilday			
	interceptor				
	failing				
	with				
	redirect-action				
WW-1429	Date	tm_jee Konstantin Pribluda	Resolved FIXED	Mar 12, 2007	Mar 13, 2007
	picker				
	ceases to				
	work				
	when				
	inside				
	ajax div				
WW-1428	Changed	tm_jee tm_jee	Resolved FIXED	Mar 10, 2007	Mar 10, 2007
	Freemarker's				
	buildScopesHashModel				
	method				
	from				
	public				
	visibility				
	to				
	protected				
WW-1357	using	Tom han weyn	Resolved FIXED	Sep 20, 2006	Mar 03, 2007
	session-to-Schneider				
	interceptor				
	generates				
	FreeMarker				
	template				
	error				
WW-1426	Tab	tm_jee tm_jee	Resolved FIXED	Mar 10, 2007	Mar 10, 2007
	panel				
	generates				
	js_error				
	on firefox				
	1.5 on				
	first load				
WW-1166	AJAX	tm_jee Jason Jones	Resolved FIXED	Feb 08, 2006	Mar 14, 2007
	submit				
	button				
	with				
	action				
	parameter				
	does not				
	submit				
	with				

action in the query parameters in Firefox 1.5				
WW-1392DWR	tm_jee	scud	Resolved CANNOT REPRODUCE	Dec 06, 2006 Mar 18, 2007
validate does not support i18n validate message?				
WW-1431Support	tm_jee	tm_jee	Resolved FIXED	Mar 20, 2007 Mar 20, 2007
non-String attributes to freemarker JSP tag extensions				
WW-1432Anchor	tm_jee	tm_jee	Resolved FIXED	Mar 20, 2007 Mar 20, 2007
tag inserts newline at end of tag				
WW-1433Client-side	tm_jee	tm_jee	Resolved FIXED	Mar 21, 2007 Mar 21, 2007
js validation doesn't work (see QuizClient example)				
WW-1435ww:set	tm_jee	tm_jee	Resolved FIXED	Mar 23, 2007 Mar 23, 2007
tag should allow value of body content to be used				
WW-1437ShowCase	tm_jee	Rainer Hermanns	Resolved FIXED	Mar 25, 2007 Mar 27, 2007
problem with sessionInvalidation feature				
WW-1439Rich_text	tm_jee	tm_jee	Resolved FIXED	Mar 28, 2007 Mar 29, 2007
editor image upload does not handle same				

image					
filename					
after it is					
uploaded					
for the					
2nd time					
or more					
WW-1438Rich text	tm_jee	tm_jee	Resolved FIXED	Mar 28, 2007	Mar 29, 2007
editor					
image					
upload					
does not					
handle					
invalid					
url path					
(windows)					
WW-1444ComponentTagSupport	tm_jee	tm_jee	Resolved FIXED	Apr 02, 2007	Apr 02, 2007
doEndTag()					
method					
doesn't					
take into					
account					
WW					
bean's					
end(...)					
method's					
result					
WW-1440Multiple	tm_jee	Richard Wallace	Resolved FIXED	Mar 28, 2007	Apr 06, 2007
forms					
cause					
conflicting					
customOnSubmit()					
functions					
to be					
created					
WW-1430Conversion	tm_jee	Perry Jeung	Resolved NOT A PROBLEM	Mar 14, 2007	Apr 08, 2007
error					
does not					
repopulate					
List fields					
WW-1445HTTP	Rainer Hermanns	Andrew Williams	Closed NOT A PROBLEM	Apr 07, 2007	Apr 11, 2007
Status					
404 -					
result					
'null' not found					
WW-1447Client-side	tm_jee	tm_jee	Resolved FIXED	Apr 11, 2007	Apr 11, 2007
js					
validation					
doesn't					
work for					
theme					
css xhtml					
when					

labelposition is top					
WW-1448 Quickstart	tm_jee	tm_jee	Resolved	FIXED	Apr 12, 2007
refuse to run in jdk6					Apr 12, 2007
WW-1441 Unable to test application using Tomcat 5 if computer is offline(doesn't have internet connection).	Rainer Marcos	HermannsMaia	Closed	CANNOT REPRODUCE	Mar 30, 2007
WW-1446 Issues with showcase before 2.2.5 release	tm_jee	tm_jee	Closed	FIXED	Apr 11, 2007
WW-1378 NullPointer in SiteGraph	Unassigned	Andreas Gabel	Resolved	FIXED	Nov 17, 2006
WW-1358 ww:doubleon and other doubleon methods not implemented	Rainer J. Hermanns	Longman	Resolved	FIXED	Sep 20, 2006
WW-1421 TokenAction - token functionality from a text link	Tom Interce Matt	SchneiderJohnston	Resolved	WON'T FIX	Jan 30, 2007
WW-1359 ww:doubleon uses first select's parameters for second select and ignores double's parameters	Rainer J.	HermannsLongman	Resolved	FIXED	Sep 20, 2006
WW-1367 Update the calendar lang file	Philip Luppens	Dieter van Baarle	Resolved	FIXED	Oct 18, 2006
					Feb 09, 2007

for					
language					
French					
WW-1365 WebWork Request Neiber shows up SchneiderCannon as bottleneck in profiling		Resolved FIXED	Oct 09, 2006	Feb 02, 2007	
WW-1362 Wrong Rainer Frederic case-sensitiv HermannLeitenberger with DateFormatter-Bean (or ww:bean-Tag) for format-param	Closed	NOT A PROBLEM	Oct 01, 2006	Oct 20, 2006	
WW-1160 Remote Rainer Adam form HermannCuper submit on Enter	Closed	WON'T FIX	Feb 07, 2006	Mar 09, 2007	
WW-1436 Relative Rainer Rafael path in HermannTorres result when invoking an action by DWRAction	Resolved	WON'T FIX	Mar 24, 2007	Mar 25, 2007	
WW-497 A tag to Patrick Aaron dump the LightbodyHeld valuestack would be useful for development	Resolved	IMPLEMENTED	Feb 10, 2004	Feb 15, 2007	
WW-1373 Export to Unassigned Plotr RTF for Sokoowski JasperReports	Resolved	FIXED	Nov 02, 2006	Feb 02, 2007	
WW-1375 Unused Tom Jasper text SchneiderRosenberg message lookup	Resolved	FIXED	Nov 07, 2006	Feb 03, 2007	
WW-1377 Backport: tm_jee Philip optiontransferselect Luppens tag does not honour doubleCssClass	Resolved	IMPLEMENTED	Nov 19, 2006	Nov 14, 2006	

Projects Using WebWork

This page last changed on Mar 12, 2007 by [tm_jee](#).

- [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
- [Cycling Gear](#) - Cycling Info Site
- [DriveNow](#) - last minute Australian 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.
- [Fathead Gear](#) \ Ecommerce App
- [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.
- [InfoQ](#) - Tracking change and innovation in the enterprise software development community
- [CableVisor](#) - A unique cable network monitoring and collaboration tool.
- [Linkomatic](#) - A link exchange system and it's FREE

Struts 2.0 Information

This page last changed on Aug 22, 2006 by [phil](#).

After the release of WebWork 2.2.2, the WebWork and Struts communities began a merge that will eventually produce Struts 2. You can find out more about this merger, as well as the latest status, by visiting the [project home page](#) or the [wiki](#).

Testimonials

This page last changed on Sep 29, 2006 by [phil](#).

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>

<http://javajmc.blogspot.com/2006/09/lightweight-java-development-with.html>

WebWork Overview

This page last changed on Feb 24, 2006 by [phil](#).

WebWork is a powerful web-based MVC framework built on top of a command pattern framework API called [XWork](#). The true power of Webwork is its underlying concept of simplicity and interoperability. Using WebWork will help minimize code and allow developers to concentrate more on business logic and modeling, rather than the plumbing often required when building web-based applications.

Features

- A flexible [Validation](#) framework allowing you to decouple validation rules from your action code.
- [Type Conversion](#) allowing you to easily convert objects from one class to another, solving one of the most tedious efforts when creating web apps.
- A powerful **Expression Language** based on [OGNL](#) allowing dynamic object graph traversal and method execution and transparent access to properties from multiple beans using a ValueStack. Webwork also has the ability to use [JSTL](#).
- [Inversion of Control](#) integration that manages component lifecycle and dependencies without the need to build registry classes that clients must call to obtain a component instance. WebWork recommends [Spring](#) for IoC.
- Reusable [Tags](#) that allow for easy and reusable component-oriented web development using [Themes and Templates](#)
- Advanced [Interceptors](#) that provide for various rich functionality, including preventing multiple form submissions and executing long running queries in the background.
- Hierarchical and pluggable support for [Internationalization](#).
- Easy integration with third party software including [Hibernate](#), [Spring](#), [Sitemesh](#), and [JSTL](#).
- Support for many view technologies such as [JSP](#), [FreeMarker](#), and [JasperReports](#).
- Modular [Configuration Files](#) using packages and namespaces to manage hundreds of actions.
- Advanced AJAX features provided by the [ajax theme](#).

About XWork

This page last changed on Jan 03, 2007 by [phil](#).

- [Upgrading XWork](#)
 - [Upgrading from 1.0](#)
 - [Upgrading from 1.0.1](#)
 - [Upgrading from 1.0.2](#)
 - [Upgrading from 1.0.3](#)
 - [Upgrading from 1.0.4](#)
 - [Upgrading from 1.1](#)
- [XWork 1.0.1](#)
- [XWork 1.0.2](#)
- [XWork 1.0.3](#)
- [XWork 1.0.4](#)
- [XWork 1.0.5](#)
- [XWork 1.0.6](#)
- [XWork Press Releases](#)
 - [1.0.1 Press Release](#)
 - [1.0.2 Press Release](#)
 - [1.0.3 Press Release](#)
 - [1.0.4 Press Release](#)
 - [1.0.5 Press Release](#)
 - [1.1.2 Press Release](#)
 - [1.2.0 Press Release](#)
 - [1.2.2 Press Release](#)

Upgrading XWork

This page last changed on Jan 03, 2007 by [phil](#).

- [Upgrading from 1.0](#)
- [Upgrading from 1.0.1](#)
- [Upgrading from 1.0.2](#)
- [Upgrading from 1.0.3](#)
- [Upgrading from 1.0.4](#)
- [Upgrading from 1.1](#)

Upgrading from 1.0

This page last changed on Jan 03, 2007 by [phil](#).

Upgrading to XWork 1.0.1 from 1.0 involves very little work. All you need to do is copy over the new xwork-1.0.1.jar in replace of xwork-1.0.jar and make sure that the new XW:Dependencies are all in place.

Upgrading from 1.0.1

This page last changed on Jan 03, 2007 by [phil](#).

Upgrading to XWork 1.0.2 from 1.0.1 involves very little work. All you need to do is copy over the new xwork-1.0.2.jar in replace of xwork-1.0.1.jar and make sure that the new XW:Dependencies are all in place.

Upgrading from 1.0.2

This page last changed on Jan 03, 2007 by [phil](#).

Upgrading from 1.0.2 is as simple as dropping in the new jar. There are no new dependencies or code changes. This release is simply a re-release of 1.0.2 with the correct packaging.

Upgrading from 1.0.3

This page last changed on Jan 03, 2007 by [phil](#).

Upgrading to XWork 1.0.4 from 1.0.3 involves very little work. All you need to do is copy over the new xwork-1.0.4.jar in replace of xwork-1.0.3.jar and make sure that the new XW:Dependencies are all in place.

Upgrading from 1.0.4

This page last changed on Jan 03, 2007 by [phil](#).

Upgrading to XWork 1.0.5 from 1.0.4 is mostly backwards compatible. However, there is one change you should be aware of:

ParametersInterceptor changes

ParametersInterceptor has been modified to not allow any parameter names to be used that contain any special OGNL characters, such as "#", ",", and "=" . This fixes a security hole, but if you were depending on it we recommend that you figure out an alternative way to do what you are doing rather than depend on this vulnerability. Upgrading to 1.0.5 fixes this issue.

Upgrading from 1.1

This page last changed on Jan 03, 2007 by [phil](#).

These are issues a developer using XWork 1.1 will need to know when migrating to XWork 1.2:

- Configuration errors fail-fast - When XWork encounters errors loading xwork and validator configuration files, it will throw an exception rather than logging the error and continue processing. The goal is that errors are more apparent to the developer and not buried in a log file somewhere, but it does mean XWork is less tolerant. Please ensure all the appropriate files load correct and report any exceptions thrown that you feel shouldn't interrupt processing.

XWork 1.0.1

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.1

Key Changes

- Introduction of an ObjectFactory that provides for easy integration to libraries like Spring and Pico
- Added actionMessages support – just like errorMessages but not counted as an error
- Major performance improvements with Ognl 2.6.5

Changelog

OpenSymphony JIRA (19 issues)		
T	Key	Summary
	XW-188	Correct logging level in DefaultActionInvocation.invokeAction(..)
	XW-183	NPE thrown when trying to set a sub-property of a property that doesn't exist
	XW-182	NPE thrown by LocalizedTextUtil.findText
	XW-167	XWorkBasicConverter conversion from Date to String is not localized
	XW-166	[Patch] Improve support for ModelDriven interface
	XW-165	[PATCH] VisitorFieldValidator not setting fieldError correctly
	XW-164	[PATCH] TypeConverter should check class hierarchy for conversion properties
	XW-163	[PATCH] Added some javadocs to Interceptors
	XW-162	TypeConverter created by ObjectFactory
	XW-161	Replaceable ObjectFactory for creating framework objects to allow easier integration with IoC containers
	XW-160	Add infinite recursion detection to the ChainingInterceptor
	XW-159	XWork build is broken
	XW-158	Email and URL Validators adding error messages for empty fields
	XW-157	Further Optimize Validator Lookup in ActionValidatorManager
	XW-156	add actionMessages support

XW-155	NPE thrown when invalid method is looked up
XW-117	Additional methods to determine if errors using JSTL
XW-56	Add localization support to XWorkConverter
XW-23	Investigate using runtime attributes to configure interceptors

XWork 1.0.2

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.2

Key Changes

- Added an **xwork-default.xml** configuration file that can be used as a standard starting place for non-web related XWork deploys (such as in Quartz)
- Localized text for collections can now be referenced in the form **someArray[XW:*].someField**, where * provides a "catch-all" for any index element
- Text retrieved using the default TextProvider now looks attempts to look for messages in the property files of child objects (if applicable) if none can be found.
- Field-level validation can be short-circuited**, meaning that you can stop the validation chain right away if you wish. See XW:Validation Framework for details.
- Simple **type conversion, such as int and float, now use the Locale**. This will cause the correct use of "," and "." depending on the locale.
- Much improved handling for type conversion of elements in collections.

Changelog

OpenSymphony JIRA (15 issues)		
T	Key	Summary
	XW-210	Make default type conversion message a localized text that can be overridden
	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's short circuit behaviour
XW-179	Optimise OgnlUtil.copy method
XW-172	XWorkBasicConverter doesn't care about the current locale

XWork 1.0.3

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.3

Key Changes

- Fixed critical problem where xwork-validator-1.0.dtd wasn't in the distribution.

Changelog

OpenSymphony JIRA (1 issues)		
T	Key	Summary
	XW-214	xwork-validator-1.0.dtd not in xwork.jar

XWork 1.0.4

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.4 Release Notes

Key Changes

This release primarily involves a few internal tweaks and API convenience methods. It is backwards compatible with 1.0.3 and should have no impact when upgrading.

Changelog

OpenSymphony JIRA (5 issues)		
T	Key	Summary
	XW-240	Added convenience methods to ComponentManager
	XW-220	OgnlValueStack.findValue doesnt use custom converter when converting to String.class
	XW-219	xwork.xml should allow no namespaces
	XW-215	upload newest xwork jars to ibiblio servers
	XW-174	ObjectFactory requires same action to return same classname

XWork 1.0.5

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.5 Release Notes

Key Changes

- Moderate security vulnerability resolved
- Configuration defaults to ActionSupport for the action if you don't define a class
- Configuration defaults to "success" for the result if you don't define a name
- Minor fixes for i18n-related issues

Migration Notes

Version	Description	Old Code	New Code
1.0.4 and below	Parameters interceptor behavior changed		

Changelog

OpenSymphony JIRA (7 issues)		
T	Key	Summary
	XW-254	Security problem with ParametersInterceptor
	XW-244	Default action class and result name
	XW-241	Allow ObjectFactory impls that support Actions without no-arg constructors
	XW-225	Whitespace in xml configuration
	XW-224	XMLConfigurationProvider should not fail if it cannot load a result type
	XW-222	ognlException while setting property 'fieldName'
	XW-180	Validation XML file loading does not work with CGLIB proxy class

XWork 1.0.6

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.6 Release Notes

Key Changes

- Reworked instantiation of Validators, such that they are now thread safe (no changes visible outside of the framework were made).
- Distribution now includes a source zip which can be used by your IDE to allow you to browse the source.

Migration Notes

Version	Description	Old Code	New Code
---------	-------------	----------	----------

Changelog

OpenSymphony JIRA (0 issues)		
T	Key	Summary

XWork Press Releases

This page last changed on Mar 25, 2007 by [phil](#).

Here you will find all the press releases for XWork. Each press release should include the [About XWork](#) page at the top of it, which you don't need to worry about as long as you use the standard Press Release template.

- [1.2.2 Press Release](#)
- [1.2.0 Press Release](#)
- [1.1.2 Press Release](#)
- [1.0.5 Press Release](#)
- [1.0.4 Press Release](#)
- [1.0.3 Press Release](#)
- [1.0.2 Press Release](#)
- [1.0.1 Press Release](#)

1.0.1 Press Release

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.1 Released

The [OpenSymphony](#) group is proud to announce the release of XWork 1.0.1. This new release paves the way for future versions of WebWork as well as new products such as "SwingWork" and other concepts that are thought up every day by OpenSymphony's active community.

- XWork: <http://www.opensymphony.com/xwork>
- Release notes and changes:
<http://www.opensymphony.com/xwork/wikidocs/Release%20Notes.html>
- Documentation: <http://www.opensymphony.com/xwork/wikidocs/Documentation.html>

New Features

The biggest addition to XWork is the new ObjectFactory which allows for tight integration to other projects such as Spring and Pico. Not only that, the ObjectFactory is very useful to integrate XWork in to your existing project infrastructure. Documentation (both reference manuals and JavaDocs) have been greatly improved as well.

Bug Fixes

There have been several important bug fixes, the most important is a performance-related bug fix that makes the expression language magnitudes faster. This also solved an extremely rare problem where some applications would lock up.

Special Thanks

Special thanks go out to Bill Lynch of [Jive Software](#) for helping with all the added JavaDocs. Mark Woon has also done a superb job assisting with bug fixes, new features, and documentation.

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

1.0.2 Press Release

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.2 Released

The [OpenSymphony](#) group is proud to announce the release of XWork 1.0.2. This release adds several new features and bug fixes in the areas of validation and type conversion.

- XWork: <http://www.opensymphony.com/xwork>
- Release notes and changes:
<http://www.opensymphony.com/xwork/wikidocs/Release%20Notes%20-%201.0.2.html>
- Documentation: <http://www.opensymphony.com/xwork/wikidocs/Documentation.html>

Special Thanks

Special thanks to Mark Woon for providing much of the code and documentation in this release. Mark has become one of the core developers for XWork and WebWork in the past couple of months and we expect to see much more come from him in the future.

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

1.0.3 Press Release

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.3 Released

The [OpenSymphony](#) group is proud to announce the release of XWork 1.0.3.

This release is a minor repackaging of 1.0.2, which was missing xwork-validator-1.0.dtd. No code changes are in this release, so upgrading is 100% safe.

- XWork: <http://www.opensymphony.com/xwork>
- Release notes and changes:
<http://www.opensymphony.com/xwork/wikidocs/Release%20Notes%20-%201.0.3.html>
- Documentation: <http://www.opensymphony.com/xwork/wikidocs/Documentation.html>

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

1.0.4 Press Release

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.4 Released

The [OpenSymphony](#) group is proud to announce the release of XWork 1.0.4. This release contains minor internal bug fixes and tweaks to the API. It is completely backwards compatible with version 1.0.3.

- XWork: <http://www.opensymphony.com/xwork/>
- Release notes and changes:
<http://www.opensymphony.com/xwork/wikidocs/Release%20Notes%20-%201.0.4.html>
- Documentation: <http://www.opensymphony.com/xwork/wikidocs/Documentation.html>
- Download: <https://xwork.dev.java.net/files/documents/709/8836/xwork-1.0.4.zip>

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

1.0.5 Press Release

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.0.5 Released

The [OpenSymphony](#) group is proud to announce the release of XWork 1.0.5.

This release fixes a moderate security vulnerability, as well as a few minor configuration improvements.

- XWork: <http://www.opensymphony.com/xwork/>
- Release notes and changes:
<http://www.opensymphony.com/xwork/wikidocs/Release%20Notes%20-%201.0.5.html>
- Documentation: <http://www.opensymphony.com/xwork/wikidocs/Documentation.html>
- Download: <https://xwork.dev.java.net/files/documents/709/9668/xwork-1.0.5.zip>

Special Thanks

A very special thanks to Aleksandr Shneyderman for discovering the security vulnerability fixed in this version.

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

1.1.2 Press Release

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.1.2 Released

The [OpenSymphony](#) group is proud to announce the release of XWork 1.1.2. This release is a bugfix release with some new improvements.

- Parameterized interceptors
- Java5 support improvements
- Javadoc improvements
- Localization and I18n improvements
- New DoubleRangeFieldValidator
- Default action support for packages
- Improved type conversion

For a complete list of all the changes, please refer to the [complete changelog](#)

OpenSymphony JIRA (41 issues)										
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
	XW-366	Improve log message in LocalizedTextUtil	Rainer Schava	HermannEugene		Closed	FIXED	Mar 21, 2006	Mar 21, 2006	
	XW-363	DefaultTextUtil does not parse default message with a MessageFormat	Rainer	Rainer		Closed	FIXED	Mar 17, 2006	Mar 17, 2006	
	XW-360	Setting allowed and blocked parameters for ParameterFilterInterceptor is not working as expected	Rainer Schava	HermannEugene		Closed	FIXED	Mar 16, 2006	Mar 21, 2006	
	XW-359	ConfigurationManager jee clearConfigurationProviders & destroyConfiguration				Resolved	FIXED	Mar 15, 2006	Mar 15, 2006	

	should call destroy appropriately						
XW-358	Translation with symbols ' and \	Rainer Schava Hermanns Eugene		Closed	FIXED	Mar 07, 2006	Mar 15, 2006
XW-357	Default action support is not existent	Andres March	Konstantin Pribluda	Closed	IMPLEMENTED	Mar 06, 2006	Mar 18, 2006
XW-356	Another new unit test	Claus Ibsen	Claus Ibsen	Closed	FIXED	Mar 05, 2006	Mar 07, 2006
XW-355	Some more unit tests and fixed a potential bug in TextProviderSupport + polished javadoc	Rainer Hermanns	Claus Ibsen	Closed	FIXED	Mar 04, 2006	Mar 04, 2006
XW-354	Again some improved unit tests	Rainer Hermanns	Claus Ibsen	Closed	FIXED	Mar 04, 2006	Mar 04, 2006
XW-353	Class javadoc improved for util package	Rainer Hermanns	Claus Ibsen	Closed	FIXED	Mar 04, 2006	Mar 04, 2006
XW-352	Unit test of LoggingInterceptor is not executed	Rainer Hermanns	Claus Ibsen	Closed	FIXED	Mar 04, 2006	Mar 04, 2006
XW-351	Update clover.jar to newest version v1.3.12	Rainer Hermanns	Claus Ibsen	Closed	FIXED	Mar 04, 2006	Mar 07, 2006
XW-350	Class javadoc improved	Rainer Hermanns	Claus Ibsen	Closed	FIXED	Mar 03, 2006	Mar 04, 2006
XW-349	Clover report without log	Rainer Hermanns	Claus Ibsen	Closed	FIXED	Mar 03, 2006	Mar 04, 2006

	debug						
XW-348	Some Rainer Claus		Closed	FIXED	Mar 01, 2006	Mar 02, 2006	
	improved Hermann Ibsen						
	unit tests						
	of						
	interceptors						
XW-347	I18nInterceptor Rainer Claus		Closed	FIXED	Mar 01, 2006	Mar 02, 2006	
	is not Hermann Ibsen						
	unit						
	tested						
XW-346	Missing Rene Claus		Closed	FIXED	Mar 01, 2006	Mar 01, 2006	
	copyright Gielen Ibsen						
	in some						
	recent						
	submitted						
	javafiles						
XW-345	PrepareInterceptor Rene Claus		Resolved	FIXED	Feb 27, 2006	Mar 01, 2006	
	is not Gielen Ibsen						
	unit						
	tested						
XW-344	ValidatorRetry Rene Claus		Closed	FIXED	Feb 27, 2006	Mar 01, 2006	
	= Gielen Ibsen						
	potential						
	loss of						
	stacktrace						
XW-343	RegexFieldValidator Rene Claus		Closed	FIXED	Feb 27, 2006	Mar 01, 2006	
	is not Gielen Ibsen						
	unit						
	tested						
XW-342	DoubleRangeValidator Rene Claus		Resolved	FIXED	Feb 26, 2006	Feb 26, 2006	
	configuration Gielen Gielen						
	should						
	not be						
	locale						
	aware for						
	the given						
	double						
	parameters						
XW-339	typeconversion Rainer Tuomas		Closed	FIXED	Feb 13, 2006	Mar 02, 2006	
	annotation Hermann Karkkainen						
	key						
	defaults						
	to empty						
	string						
XW-338	Incorrect Alexandru Michal		Resolved	FIXED	Feb 13, 2006	Mar 10, 2006	
	handling Popescu Karwanski						
	of						
	localized						
	numbers						
	while						
	converting						
	to						
	primitive						
	int						

XW-337	Add @Key, @Element and @CreateIfNull annotations	Rainer Hermanns	Rainer Hermanns	Closed	FIXED	Feb 09, 2006	Feb 15, 2006
XW-336	XWork type conversion of Milisecond support	Rainer Hermanns	katsumi takahashi	Resolved	FIXED	Feb 08, 2006	Feb 08, 2006
XW-335	@StringLength creates log-warnings when not setting property maxLength	Rainer Hermanns	Peter Westlin	Resolved	FIXED	Feb 07, 2006	Feb 08, 2006
XW-332	Chain Result throws Exception when you use a actionName with a method	Alexandru Popescu	Alberto Vilches	Resolved	FIXED	Feb 02, 2006	Mar 03, 2006
XW-314	Create ParameterFilterInterface	Unassigned Rainer	Gabriel Zimmerman	Closed	FIXED	Dec 22, 2005	Feb 09, 2006
XW-294	Downgrade error logging of missing result classes	Rainer Hermanns	David Croft	Closed	FIXED	Jul 25, 2005	Mar 02, 2006
XW-291	configuration improvement to detect misconfigurations (different package with same name)	Rainer Hermanns	Luigi R. Iiggiano	Closed	FIXED	Jun 30, 2005	Mar 04, 2006
XW-287	Cached validators hang on to their old contexts	Rainer Hermanns	Tom Davies	Closed	FIXED	May 25, 2005	Mar 04, 2006
XW-286	ognl parsing	Rainer Hermanns	Mike Mosiewicz	Closed	FIXED	May 19, 2005	Feb 08, 2006

	of values in StaticParametersInterceptor				
XW-281	Query AlexandruMike String Popescu Mosiewicz containing large numbers causes exception	Resolved	FIXED	Apr 29, 2005	Mar 10, 2006
XW-280	Configura Reiner John should Hermanns Patterson distinguish an empty value from a missing value	Closed	FIXED	Apr 23, 2005	Mar 04, 2006
XW-277	Shouldn't Rainer Grégoire Localized Hermanns Joseph log level be raised?	Closed	FIXED	Apr 19, 2005	Feb 20, 2006
XW-269	Conversion Reiner Erik Logi bug Hermanns when propertyName contains single quotes	Resolved	NOT A PROBLEM	Feb 28, 2005	Feb 08, 2006
XW-252	TextProvider Rainer Alex should Hermanns Shneyderman have more methods on its interface	Closed	FIXED	Dec 10, 2004	Feb 15, 2006
XW-248	OgnlUtil.class Rainer sutter2k corrupts Hermanns cglib proxied actions	Closed	FIXED	Dec 04, 2004	Mar 02, 2006
XW-223	Improve Rainer Jeroen "No Hermanns van object in Vianen the CompoundRoot has a property named 'bar" exception	Closed	FIXED	Sep 16, 2004	Mar 04, 2006
XW-216	Create a Rainer Ricardo DoubleRank Hermanns Michael Individual Select	Closed	FIXED	Sep 09, 2004	Feb 08, 2006

[XW-212](#) [serializable](#)
[configuration](#)

Resolved FIXED Aug 24, 2004 Feb 09, 2006

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

1.2.0 Press Release

This page last changed on Jan 03, 2007 by [phil](#).

XWork 1.2.0 Released

The [OpenSymphony](#) group is proud to announce the release of XWork 1.2.0. This release is a bugfix release with some new improvements.

- Type converters on nested properties
- Java5 support improvements
- Line-precise error reporting

For a complete list of all the changes, please refer to the [complete changelog](#)

OpenSymphony JIRA (37 issues)										
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
	XW-400	i18n and ww:text tag can throw an exception under certain conditions	Rainer Hermanns	Nick Hill		Resolved	FIXED	Jul 12, 2006	Jul 17, 2006	
	XW-396	Alter tm_jee prepareInterceptor to call prepare{MethodName} when doing action!methodName	tm_jee	tm_jee		Resolved	FIXED	Jul 03, 2006	Jul 04, 2006	
	XW-386	Xwork + SpringIOC. Xwork don't creates the result calling appContext.getBean("myResult").	tm_jee	Fabio Falci		Resolved	FIXED	Apr 17, 2006	Jul 22, 2006	
	XW-385	type conversion of array doesn't work	Unassigned	tm_jee		Resolved	WON'T FIX	Apr 16, 2006	May 21, 2006	
	XW-384	DefaultAction Andres calls Rife every	Andres March	Andres March		Resolved	FIXED	Apr 13, 2006	Apr 13, 2006	

	time						
XW-383	Validation configuration errors should fail-fast	Don Brown	Don Brown	Resolved	FIXED	Apr 10, 2006	Apr 10, 2006
XW-380	Debug String is output to System.out.println()	Rainer Hermanns	Corby Page	Resolved	FIXED	Apr 02, 2006	May 10, 2006
XW-379	Line-precision error reporting	Don Brown	Don Brown	Resolved	FIXED	Apr 02, 2006	Apr 04, 2006
XW-378	Make stack lookup failure better visible in devMode	Rene Gielen	Rene Gielen	Closed	FIXED	Mar 30, 2006	Mar 30, 2006
XW-377	com.opencafe.babel.util.LocalizedText generate warning when message is the same as key	Ibsen	Ibsen	Resolved	FIXED	Mar 30, 2006	Apr 09, 2006
XW-376	Validation ClassSupport - synchronized - missing some methods	Claus Ibsen	Claus Ibsen	Closed	FIXED	Mar 30, 2006	Apr 04, 2006
XW-375	Conflicting validators	Ibsen	Patrick Lightbody	Resolved	FIXED	Mar 28, 2006	Jun 11, 2006
XW-374	TimerInterface - support setting log category	Ibsen	Ibsen	Resolved	IMPLEMENTED	Mar 27, 2006	Apr 04, 2006
XW-373	xwork conversion validator should allow previous value to be displayed upon error in conversion	tm_jee	tm_jee	Resolved	FIXED	Mar 27, 2006	Apr 02, 2006
XW-372	Date	Rainer	tm_jee	Resolved	FIXED	Mar 27,	May 10,

	Conversion	Hermanns			2006	2006
	should be null instead of conversion error when http request is an empty string					
XW-371	invalid mail address validation failed	tm_jee katsumi takahashi	Resolved FIXED	Mar 23, 2006	Apr 27, 2006	
XW-365	Exception Mapping Interceptor should allow logging the exception	Ibsen	Closed IMPLEMENTED	Mar 21, 2006	Apr 04, 2006	
XW-340	Parameters for interceptors does not evaluated using ognl stack	Alexandru Schava Popescu Eugene	Closed NOT A PROBLEM	Feb 21, 2006	Jun 22, 2006	
XW-320	Add tiger and Lightbody	Rainer Hermanns	Resolved FIXED	Jan 13, 2006	Jul 10, 2006	
	tiger-test targets to AntHill setup					
XW-297	Type Converters Can't Be Set On Nested Properties	Patrick Doug Lightbody Seifert	Resolved FIXED	Feb 10, 2004	Jul 24, 2006	
XW-296	Multiple components with same <class> but different <enabler> throws exception	Patrick Matthew Lightbody Denner	Resolved WON'T FIX	Apr 22, 2004	Jun 23, 2006	

XW-288	OGNL expression cache to use IdentityHashMap	Alexandru John Popescu Patterson	Closed	WON'T FIX	Jun 09, 2005	Jun 26, 2006
XW-282	NPE thrown when trying to set a sub-property of a property that exists	Alexandru Ganesh Popescu Bhattachan	Closed	CANNOT REPRODUCE	Apr 29, 2005	Jun 22, 2006
XW-279	ognlValue too intolerant of NPEs	Alexandru Kenny Popescu MacLeod	Closed	WON'T FIX	Apr 22, 2005	Jun 22, 2006
XW-278	Expression validation on a non-alias file get validated on alias file as well, thus giving 2 error messages	Rainer Steve Hermann Loh	Resolved	CANNOT REPRODUCE	Apr 21, 2005	May 21, 2006
XW-271	Child property determination for localization	Alexandru Andrei Popescu Ivanov	Resolved	FIXED	Mar 27, 2005	Jul 29, 2006
XW-270	Interceptor supports IoC	Rainer Gang Hermann Chan	Closed	NOT A PROBLEM	Mar 01, 2005	Jun 20, 2006
XW-260	xWork does not report syntax errors in OGNL expression	Alexandru Andriy Popescu Palamarchuk	Closed	IMPLEMENTED	Mar 12, 2005	Jun 26, 2006
XW-249	Enable external-ref elements on interceptors	Rainer Alex Hermann Shneyderman	Closed	WON'T FIX	Dec 07, 2004	Apr 11, 2006
XW-233	Interceptor initialized twice	Alexandru Jeroen Popescu van Vianen	Closed	CANNOT REPRODUCE	Oct 15, 2004	Jun 22, 2006

XW-208	short-circuiting of action-level validators before field-level validators	Alexandru Popescu Hernandez	Closed	IMPLEMENTED	Jun 11, 2004	Jun 22, 2006
XW-201	visitor validation doesn't work if the object to be validated is null	Jason Carreira Hernandez	Resolved	WON'T FIX	Jun 18, 2004	May 21, 2006
XW-184	ability to pass parameters to the message key in the validation system	Alexandru Popescu Hernandez	Closed	WON'T FIX	May 27, 2004	Jun 22, 2006
XW-176	Create an InterceptorManager to cache interceptors	Alexandru Popescu Cannon-Brookes	Closed	WON'T FIX	May 06, 2003	Jun 22, 2006
XW-169	Programmatic Configuration of XWork	Jason Carreira Simon Stewart	Resolved	FIXED	Mar 08, 2004	Jun 23, 2006
XW-108	XWorkBasicImprovement	Alexandru Popescu Mosiewicz	Closed	FIXED	Oct 15, 2003	Jun 26, 2006
XW-102	RuntimeException should be replaced with proper exception handling	Rainier Hermanns Mosiewicz	Closed	FIXED	Oct 06, 2003	Jul 13, 2006

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated together to form complete applications independently.
- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

1.2.2 Press Release

This page last changed on Mar 25, 2007 by [phil](#).

XWork 1.2.2 Released

The [OpenSymphony XWork](#) team is proud to announce the release of XWork 1.2.2, the latest release in the 1.2 branch of XWork, a command pattern framework, which forms the core of the WebWork 2 framework.

This release specifically addresses the problem related to the random NPE's that can be thrown by different XML parsers when the xwork.xml file is loaded, as well as various bugs in the annotations for validation and configuration.

For a complete list of all the changes, please refer to the [complete changelog](#)

OpenSymphony JIRA (22 issues)										
T	Key	Summary	Assignee	Reporter	Pr	Status	Res	Created	Updated	Due
	XW-506	Maven build fails on 1.2 branch	Claus Ibsen	Claus Ibsen		Closed	FIXED	Apr 09, 2007	Apr 10, 2007	
	XW-482	Unit tests failing on 1.2	Philip Luppens	Philip Luppens		Resolved	WON'T FIX	Feb 12, 2007	Mar 22, 2007	
	XW-481	XmlAttributeProvider fails to parse results because Node.equals fails	Rainer Pojman	Rainer Pojman		Closed	FIXED	Feb 11, 2007	Feb 16, 2007	
	XW-480	Annotations in package com.opensymphony.xwork.util missing @Retention	Rainer Hermanns	Kari Helenius		Closed	FIXED	Feb 09, 2007	Feb 16, 2007	
	XW-478	i18n interceptor malfunction	Rainer Hermanns	Miguel Montreal		Closed	NOT A PROBLEM	Oct 04, 2006	Mar 25, 2007	
	XW-474	Backport Date handling from XWork2 to XWork 1.2 branch	Philip Luppens	Philip Luppens		Resolved	FIXED	Feb 01, 2007	Feb 01, 2007	

XW-469	JavaDoc	Rainer	Ted	Closed	FIXED	Jan 28, 2007	Feb 16, 2007
	type	Hermann	Husted				
XW-467	Extra	tm_jee	tm_jee	Resolved	FIXED	Jan 25, 2007	Feb 02, 2007
	ognl						
	Value						
	Stack						
	setting in						
	StaticParameterInterceptor						
XW-465	XWork	Tom	tm_jee	Resolved	FIXED	Jan 22, 2007	Feb 04, 2007
	failed to	Schneider					
	execute						
	action						
	that is a						
	java.lang.reflect.Proxy						
XW-458	Java 5.0	Rainer	Jeff	Resolved	FIXED	Jan 02, 2007	Jan 02, 2007
	reference	Hermann	Bailey				
	in						
	xwork-1.2.2						
XW-457	XWork	Unassigned	Philip	Resolved	FIXED	Jan 02, 2007	Feb 04, 2007
	throws		Luppens				
	random						
	NPE's						
XW-456	Declarative	Rainer	Tom	Closed	DUPLICATE	Dec 29, 2006	Mar 25, 2007
	validation	Hermann	Schneider				
	of bean						
	arrays						
XW-452	declarative	tm_jee	tm_jee	Resolved	FIXED	Dec 02, 2006	Mar 25, 2007
	validation						
	should be						
	able to						
	handle						
	validation						
	of list						
XW-442	NPE in	Rainer	Andrei	Resolved	FIXED	Nov 27, 2006	Nov 29, 2006
	visitor	Hermann	Ivanov				
	validator						
XW-439	clean up	tm_jee	tm_jee	Resolved	FIXED	Nov 17, 2006	Nov 18, 2006
	of						
	ConfigurationManager						
	is						
	incomplete						
XW-423	Intercept	tm_jee	tm_jee	Resolved	FIXED	Oct 04, 2006	Oct 04, 2006
	interceptor						
	param						
	overriding						
	is not						
	isolated						
	to per						
	action						
XW-421	XWork	Rainer	Jonathan	Closed	FIXED	Oct 04, 2006	Nov 16, 2006
	Annotation	Hermann	Gerrish				
	bugs?						
XW-417	Domain	Rainer	Jim	Closed	NOT A PROBLEM	Sep 24, 2006	Oct 07, 2006
	model	Hermann	Horner				

	disappears from value stack after getText()					
XW-412	Some XMLParsers parse the xwork config file differently, resulting in broken action results	Rainer Hermanns	Bill Lynch	Closed	FIXED	Sep 09, 2006 Feb 16, 2007
XW-411	OgnlValueShade cannot handle multiple overrides.	OgnlValueShade	Derek Clarkson	Resolved	FIXED	Sep 07, 2006 Sep 08, 2006
XW-410	RepopulateCoercersForFieldValidationReport not restoring field values when dealing with domain objects.	RepopulateCoercersForFieldValidationReport	Derek Clarkson	Resolved	FIXED	Sep 07, 2006 Sep 08, 2006
XW-403	XWork configuration using Annotation	Rainer Hermanns	Jecki Hermanns	Closed	FIXED	Aug 08, 2006 Mar 25, 2007

About WebWork

WebWork is a leading open source Java web application framework. Developed originally by Rickard Oberg (original developer of JBoss and creator of XDoclet, among other accomplishments), WebWork aims to lower the bar for developing web applications by making the more tedious tasks of web development automated. By taking the best features from other web frameworks available today, WebWork represents a best-of-bread solution to web development created by through the feedback of an active OpenSymphony community.

WebWork is built on top of [XWork](#), a generic command pattern framework. WebWork uses the capabilities of XWork to provide the following features:

- Advanced UI components, allowing you to build complex, reusable UI components, ranging from simple text fields to advanced date pickers.
- A robust inversion of control (IoC) container that binds to the native Servlet lifecycles: request, session, and application.
- Pluggable configuration, allowing you to develop web "modules" that can easily be integrated

together to form complete applications independently.

- Complete data mapping from HTTP to Java data objects, enabling you to focus more on application development and less on tedious data conversion.
- A complete validation framework, both on the server side and client side. This lets you choose the most optimal way to ensure user input is correct before processing it.
- An advanced expression language, based on [OGNL](#), providing the most common operations usually associated with building web-based user interfaces.
- Support for integration with many popular open source projects, including: Spring, Pico, OSWorkflow, FreeMarker, Velocity, JasperReports, JFreeChart, and many more.

Continuous Build of WebWork and XWork

This page last changed on Apr 09, 2007 by [tm_jee](#).

WebWork and XWork are build continuously whenever a commit is done. The tests are run and finally a jar file is being bundled up.

The jar files of WebWork and XWork as a result of successful continuous build could be found at:-

- [Ivy Repository](http://ivyrep.opensymphony.com/) (<http://ivyrep.opensymphony.com/>)
- [Maven 1](http://maven.opensymphony.com/) (<http://maven.opensymphony.com/>)
- [Maven 2](http://maven2.opensymphony.com/) (<http://maven2.opensymphony.com/>)

This is helpful for users willing to try out the latest build that might have bug fixes critical to them instead of building from scratch.

Developers

This page last changed on Jan 03, 2007 by [phil](#).

Building WebWork

This page last changed on Jan 31, 2007 by [phil](#).

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 continuous 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.

SVN

The full source code can be found at the OpenSymphony Subversion repository:
<https://svn.opensymphony.com/svn> under the *webwork* directory.

If you want to access the latest version, be sure to check out the *webwork/trunk* directory. There are some other branches available under *webwork/branches* (such as WW 2.1.x).

We suggest using a plugin to checkout the sources and commit changes. There are several plugins available for both Eclipse (subclipse) and IntelliJ (built in, nowadays).



If anyone bothers to check out the command line commands for SVN, please put them here.

Building

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

command	description
ant init	let ivy update webwork's dependencies
ant clean	clean up webwork build
ant test	run webwork tests cases
ant compile	compile webwork source code
ant test-compile	compile webwork testcases source code
ant jar	create a jar file for webwork
ant dist	create a webwork distribution (with jar file, javadoc etc.)

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 dependencies 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 resolve 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 and its license jar file (>= 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.

Style Guide

This page last changed on Mar 21, 2006 by [phil](#).

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 Langhorne 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 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-usa
```

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 preceding "* " (asterisk-space) characters will be stripped before printing the content out. Also, the content is assumed to be 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: xxxx
    System.out.println("This is some silly code!");
    // END SNIPPET: xxxx
}
```

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 definitely be used. This section 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 definitely 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: `<xmlelement />`, `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 amateurishness 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.

Documentation

This page last changed on Mar 23, 2006 by [plightbo](#).

This page has moved [here](#).

Help

This page last changed on Jan 03, 2007 by [phil](#).

- [Chat](#)
- [FAQ](#)
- [The Vault](#)

Chat

This page last changed on Feb 22, 2006 by [phil](#).

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 21, 2006 by [phil](#).

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 Jun 15, 2005 by [plightbo](#).

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
 jcarreira 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      yup.. 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
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 - remoteclick sends a "refresh", tab header sends a "tab selected"

iroughley remote div can specify a topic to listen to and refresh 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" 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 need to utilize multiple themes for a single component?

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
iroughley     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      yup, 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-       yup
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 widget 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 the dojo:topic tag specify the topic to listen to and the topic to
publish to when an event occurs?
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 - will 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 iroughley we need a wiki page to list the ajax theme components (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/...

06-15-2005 Documentation

This page last changed on Jun 15, 2005 by [vitorsouza](#).

Attendees

- Erik Beeson
- Simon Stewart
- Jason Carrera
- Patrick Lightbody
- Jay Bose
- Vitor Souza
- Grégory 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](#)
 - Standalone 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 Grégory 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-man
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      yup
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 fragmented 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 beginners 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 also 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 beginning 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 goals.

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 the 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 its 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 correctly
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 be more 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 manual
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 +
-->| 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 Hmm... 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
```

06-15-2005 TOC Homework

This page last changed on Aug 03, 2005 by [plightbo](#).

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:	Begginer	Intermediate
Availability of Source code:	In the documentation and in the <code>src</code> folder of the distribution (ready for deploy and test).	Only in the documentation, and may not be <i>complete</i> .
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, Freemaker, 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. ?

07-04-2005 Documentation

This page last changed on Aug 03, 2005 by [plightbo](#).

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> forgot 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 briefly
```

[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> Hmm... 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 Cauchy 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> scriptlets?

[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 Freemaker 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 Freemaker 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, Freemaker, 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 Hibernate 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> ahhh
[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 style.
[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.

FAQ

This page last changed on Apr 11, 2007 by phil.



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 currently too verbose and should be broken down given that they have already been categorized (ie: Validation, Internationalization, 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](#)
- [How can I get the ServletContext](#)
- [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](#)
- [Why does WebWork fail to find some javascript](#)
- [Why does WebWork only manage to retrieve attribute from HttpSession the second time around](#)
- [Why does my setter not get called](#)
- [Why does my FileUpload not work](#)
- [Why does FileUpload ignore the file size rule specified](#)
- [Why does WebWork failed when trying to validate my xml configuration file against its dtd](#)

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](#)
- [Why does WebWork's Rich Text Editor does not render](#)
- [Why doesn't WebWork's If tag evaluate test="#parameters.someParam ... " properly](#)
- [Why does Url tag compound my namespace](#)
- [Some container complains about OGNL expression while parsing my jsp page](#)

- [How do I set a date into datepicker component](#)
- [How do I allow WebWork action to be executed only using WebWork Action Tag but not from direct url access from browser](#)
- [What should I do about the conflict with JSP 2.1 EL and WebWork OGNL EL](#)
- [I want my form's submit button to be displayed differently eg. besides my text field](#)

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](#)
- [How do I unit test my action's validation logic](#)
- [Why does WW ignore my message when its enclosed in CDATA](#)
- [Why do some validators like stringlength ignore validation when input is empty, it worked before](#)

Internationalization

- [How do I set a global resource bundle](#)
- [How do I decouple XWork LocalizedTextUtil global resource bundle loading from servlets](#)
- [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](#)

Ajax / Dojo

- [Internet Explorer showing a prompt saying 'This page contains both secure and nonsecure items' when using dojo](#)

Portlet Support (JSR168)

- [Which portal servers are supported](#)
- [How to build the portlet war for a specific portal server](#)

Sitemesh

- [Problem getting SiteMesh to work with web.xml errorPage](#)

DisplayTag

- [WebWork failed to grab parameters of sort link generated by DisplayTag](#)

Can I access my action's Result

This page last changed on Nov 21, 2005 by [tm_jee](#).

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 Nov 05, 2005 by [digi9ten](#).

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 Nov 09, 2005 by [rgieLEN](#).

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 Jan 26, 2006 by [emolitor](#).

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 Jan 26, 2006 by [emolitor](#).

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 Jan 26, 2006 by [emolitor](#).

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 Jan 26, 2006 by [emolitor](#).

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 Feb 25, 2006 by [tm_jee](#).

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 Dec 07, 2005 by [tm_jee](#).

With the html as follows:

```

</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 Nov 09, 2005 by [rgieLEN](#).

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.ServleConfigInterceptor`

How can I get the HttpServletResponse

This page last changed on Nov 09, 2005 by [rgieLEN](#).

Method A:

`ServletActionContext.getResponse()` (works internally using a ThreadLocal)

Method B:

Have the action implements `ResponseAware` and the response will be set through `setServletResponse(HttpServletRequest)`. 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 get the ServletContext

This page last changed on Feb 13, 2007 by [phil](#).

Q: How can I get access to the ServletContext in my Action ?

A: Have your action implement com.opensymphony.webwork.util.ServletContextAware.

How can I integrate WebWork IoC in to an object that is not an action

This page last changed on Nov 05, 2005 by [digi9ten](#).

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 Nov 05, 2005 by [digi9ten](#).

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 Feb 10, 2006 by [n036939](#).

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 06, 2006 by [tmky2k](#).

```
<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 allow WebWork action to be executed only using WebWork Action Tag but not from direct url access from browser

This page last changed on Mar 12, 2007 by [tm_jee](#).

One possible solution would be to have an interceptor that checks the action context for the existence of a `ServletActionContext.PAGE_CONTEXT` variable. If it's available, the action is being executed in a page context.

Detailed discussion is [here](#)

Special thanks to Philip Luppens for contributing the idea.

How do I change the error message for invalid inputted fields

This page last changed on Nov 05, 2005 by [digi9ten](#).

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 servlets

This page last changed on Nov 05, 2005 by [digi9ten](#).

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 Nov 09, 2005 by [rgieLEN](#).

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 Nov 10, 2005 by [tm_jee](#).

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 Nov 20, 2005 by [tm_jee](#).

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 Nov 09, 2005 by [rgieLEN](#).

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 Nov 10, 2005 by [tm_jee](#).

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 require 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(ServletRequest,
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 29, 2006 by [tm_jee](#).

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 date into datepicker component

This page last changed on Dec 08, 2006 by [tm_jee](#).

The trick is to format the date correctly before sending it to the datepicker, and let datepicker what format it is supposed to be using.

```
<ww:form name="form" action="to_infinity_and_beyond" method="post">
  <ww:date name="#{new java.util.Date()}" format="dd-MM-yyyy hh:mm" id="date"/>
  <ww:datepicker value="#{#date}" showstime="#{true}" format="%d-%m-%Y %H:%M"/>
</ww:form>
```

contributed by Philip Luppens

How do I set a global resource bundle

This page last changed on Nov 10, 2005 by [plightbo](#).

In webwork.properties(as of Webwork 2.1.1),
you can now use:

```
webwork.custom.i18n.resources=global-messages
```

Several 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.ServletContextEvent)  
     */  
    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 unit test my action's validation logic

This page last changed on Oct 01, 2006 by [perfurm](#).

```
public class WebLoginActionTest extends TestCase {

    private WebLoginAction wla; //Your custom WW2 action that extends ActionSupport
    // If the WW2 action contains other complex objects (i.e. domain model objects), setUp()
    can be used to initialize those objects
    protected void setUp() throws Exception {
        wla = new WebLoginAction();
        wla.setJ_username("");
        wla.setJ_password(null);
        super.setUp();
    }

    // The WiKi shows different code for testing the validation for an action, the one below is
    proven to work
    public void testWebLoginActionValidation() throws ValidationException {
        ActionValidatorManager avm = ActionValidatorManagerFactory.getInstance();
        avm.validate(wla, "");
        Map fieldErrors = wla.getFieldErrors();

        assertTrue(wla.hasErrors());
        assertEquals(2, fieldErrors.size());
        assertTrue(fieldErrors.containsKey("j_username"));
        assertTrue(fieldErrors.containsKey("j_password"));

        System.out.println("[errors] : " + fieldErrors.toString()); // Displays validated
        fields and its associated error messages
    }
}
```

Contributor: Carlos Cajina

How do I use messages from within the validator

This page last changed on Nov 05, 2005 by [digi9ten](#).

```
<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 Feb 02, 2006 by [rainerh](#).

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 Feb 21, 2007 by [tm_jee](#).

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 {. (see javadoc API for MessageFormat)

The special chars ', } and {:

- escape ' with another ' : resulting in " (double-single quote)
- escape \ with another \: resulting in \\ (double backslash)
- enclose } with ': resulting in '}'
- enclose { with ': resulting in '{'

How to reload xwork configuration

This page last changed on Jan 12, 2006 by [victorsosa](#).

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 Nov 24, 2005 by [rainerh](#).

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 want my form's submit button to be displayed differently eg. besides my text field

This page last changed on Apr 09, 2007 by [tm_jee](#).

The idea is to make use of WebWork's theme and templating to style the button's position as one wish. For example to have the submit button besides a text field, one could do the followings

Method one

Using simple theme, which doesn't enforce a certain layout pattern like certain themes such as xhtml (using tables) and css_xhtml (using divs and spans) does.

Apply the simple theme by adding theme="simple" in the tags for the components. The following examples are in Freemarker, adjust as necessary for jsp:

```
<@ww.textfield name="foo" theme="simple"/>
<@ww.submit theme="simple"/>
```

or alternatively,

```
<@ww.form ... theme="simple"/>
<@ww.textfield name="foo" />
<@ww.submit />
</@ww.form>
```

Method two

An alternate approach would be to use the 'after' parameter that's available in most WebWork tags. An example using Freemarker would be as follows, adjust as necessary for jsp:

```
<@ww.textfield name="foo" theme="xhtml">
<@ww.param name="after">
    <@ww.submit theme="simple"/>
</@ww.param>
</@ww.textfield>
```

This will render the submit button within the same table column as the text field (since xhtml styled components using HTML table).

Relevant information

- [Themes and Templates](#)
- [Selecting Themes](#)

p/s: Special thanks to Lens and Philip Luppens for the information.

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 Nov 05, 2005 by [digi9ten](#).

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.

Internet Explorer showing a prompt saying 'This page contains both secure and nonsecure items' when using dojo

This page last changed on Aug 25, 2006 by [tm_jee](#).

The "security and nonsecure items" message in IE is being caused by an issue in Dojo. For more, see the discussion on the [WebWork forum](#).

There is a workaround available from the [Dojo mailing list](#).

In short, line 79 in dojo/src/storage/browser.js

```
storeParts.push('codebase="http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
```

should be replaced with

```
storeParts.push('codebase="https://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0"
```

Also try to search dojo.js and see if the following exists

```
_607.push("\tcodebase=\"http://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0\"")
```

if it does should be replaced with

```
_607.push("\tcodebase=\"https://download.macromedia.com/pub/shockwave/cabs/flash/swflash.cab#version=6,0,0,0\"")
```

Problem getting SiteMesh to work with web.xml errorPage

This page last changed on Dec 03, 2006 by [tm_jeetm_jeet](#).

The doFilter() method of PageFilter sets the boolean variable FILTER_APPLIED on the request to true, before executing. When an exception occurs in the call to chain.doFilter(), the execution stops. After the servlet invokes the error page, the PageFilter is invoked again, but stops when it sees that the FILTER_APPLIED attribute is already set to true.

The solution is to extend the PageFilter, catch all exceptions and set the FILTER_APPLIED variable to null. Then use this filter in the web.xml.

```
public class ErrorHandlingSiteMeshPageFilter extends PageFilter {  
  
    public void doFilter(ServletRequest request, ServletResponse rs, FilterChain chain) throws  
        IOException, ServletException {  
        try {  
            super.doFilter(request, rs, chain);  
        } catch (RuntimeException e) {  
            clearFilteredVariable(request);  
            throw e;  
        } catch(IOException e) {  
            clearFilteredVariable(request);  
            throw e;  
        } catch(ServletException e) {  
            clearFilteredVariable(request);  
            throw e;  
        }  
    }  
  
    private void clearFilteredVariable(ServletRequest request) {  
        request.setAttribute(FILTER_APPLIED, null);  
    }  
}
```

For more information of related discussion, see [here](#)

Solution contributed by Hendrikd. Thx Hendrikd.

Some container complains about OGNL expression while parsing my jsp page

This page last changed on Nov 15, 2006 by [tm_jee](#).

To solve this issue, one will have to disable EL using

```
<jsp-config>
  <jsp-property-group>
    <url-pattern>*.jsp</url-pattern>
    <el-ignored>true</el-ignored>
  </jsp-property-group>
</jsp-config>
```

in web.xml

Thanks to Richard Wallace for the answer.

WebWork failed to grab parameters of sort link generated by DisplayTag

This page last changed on Mar 12, 2007 by [tm_jeo](#).

This is due to WebWork attempting to parse the link (eg.

<http://localhost:8080/etg-webapp/secure/reports/byReviewer.action?d-4002248-s=2&etgId=2&etgId=2&d-4002248-0>

) using OGNL and OGNL treats eg. d-400248... as minus resulting in the undesired behaviour in dev mode.

The solution would be to have WebWork action implements ParameterNameAware and filter out the offending parameter.

```
public class MyAction extends ActionSupport implements ParameterNameAware {  
  
    /**  
     * This method will filter out parameters that  
     * start with "d-" followed by a numeric digit,  
     * because parameters of this form are generated by displayTag,  
     * and are treated by webwork as an invalid OGNL expressions causing  
     * webwork to throw an ognl.InappropriateExpressionException exception.  
     * Note that the exception is only thrown when webwork is set in devmode.  
     * However, to prevent this error, the ParameterNameAware interface has been  
     * implemented which requires this method.  
     * This method could be implemented more simply using java's regular expression  
     * support, but such an implementation may suffer from readability except for  
     * people who are very strong in understanding java's RegEX semantics.  
     */  
    public boolean acceptableParameterName(String parameterName){  
        boolean retVal = true;  
        if(parameterName!=null && parameterName.startsWith("d-") )  
            if( parameterName.length()>2) {  
                String thirdCharacter = parameterName.substring(2,3);  
                if(StringUtils.isNumeric(thirdCharacter)){  
                    retVal = false;  
                }  
            }  
        return retVal;  
    }  
}
```

Special thanks to Richard Wallace and Philip Brown. The detailed discussion is [here](#)

What are the default variables in the value stack

This page last changed on Nov 08, 2005 by [tm_jee](#).

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	'parameters' above
ActionContext.SESSION	com.opensymphony.xwork.ActionContext.session	'session' above
ActionContext.APPLICATION	com.opensymphony.xwork.ActionContext.application	'application' above
ActionContext.LOCALE	com.opensymphony.xwork.ActionContext.locale	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	Servlet request
ActionContext.HTTP_RESPONSE	com.opensymphony.xwork.dispatcher.HttpServletResponse	Servlet response
ActionContext.SERVLET_CONTEXT	com.opensymphony.xwork.dispatcher.ServletContext	Servlet Context object
ActionContext.COMPONENT_MANAGER	com.opensymphony.xwork.interceptor.ComponentManager	Webwork's Component manager

For further information

@see com.opensymphony.webwork.dispatcher.DispatcherUtils#createContextMap

@see com.opensymphony.xwork.interceptor.ComponentInterceptor

@see com.opensymphony.webwork.WebWorkStatics

What should I do about the conflict with JSP 2.1 EL and WebWork OGNL EL

This page last changed on Mar 15, 2007 by [tm_jee](#).



Content of this page is to be verified of its validity.

It should continue to work. Have a look at [here](#) for more info.

Which portal servers are supported

This page last changed on Feb 02, 2006 by [rainerh](#).

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 Jan 17, 2006 by [tm_jee](#).

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 Feb 25, 2006 by [tm_jee](#).

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 Jan 10, 2006 by [plightbo](#).

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 do some validators like stringlength ignore validation when input is empty, it worked before

This page last changed on Apr 11, 2007 by [phil](#).



Note

This behavior is added since WebWork 2.2.4 (in the XWork bundle that comes with it). See XW-422 for more information.

WebWork's validators are designed to be as independent of each other as possible, such that a validator like 'stringlength' should just validate the length of the input string. It will not attempt to validate an empty string as that would overlap with the 'requiredstring' validator, for example.

If we need to validate against an input that should not be empty and should have a certain length, we would apply the following validator sequence :

- 'requiredstring' validator
- 'stringlength' validator

If we need to validate against an input that should have a certain length but it doesn't matter if nothing is specified, we could apply the following validator sequence :

- 'stringlength' validator

Without the concept of 'independent responsibility' in each validator design, the second case in the above described scenarios would not be possible to achieve.

For more information about validators, have a look at [Validation](#).

Why does FileUpload ignore the file size rule specified

This page last changed on Jan 07, 2007 by [tm_jee](#).

One might want to have a look at [File Upload Interceptor](#).

When Jakarta Commons Fileupload is used, the size defined in webwork.properties will be ignored, instead define the size in FileUploadInterceptor.

Why does FreeMarker complains that there's an error in my user-directive when I used JSP Tag

This page last changed on Nov 14, 2005 by [tm_jee](#).

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 my FileUpload not work

This page last changed on Jan 07, 2007 by [tm_jee](#).

One might want to have a look at [File Upload Interceptor](#).

One possibility is that the dependent jar files are missing in the classpath. Since WebWork 2.2.5, an error will be logged on console if the dependencies are found missing even if an "input" result is not defined. Previous version might swallowed the exception. If an "input" result is defined, the error will be registered, and the "input" result page will be rendered.

Why does my setter not get called

This page last changed on Jan 04, 2007 by [pledge](#).

Why doesn't my setter get called ?

[Aaron Johnson](#) had a [problem](#) with a setter for a property that never seemed to get called. He had a getter/setter combination that did not match:

```
public void setUser(String username) {  
    ...  
}  
public User getUser() {  
    ...  
}
```

The short answer is that your getter and setter must match the property type, or OGNL will not be able to call it (due to a reflection API semantics). You can read the full explanation [here](#).

Why does Url tag compound my namespace

This page last changed on Nov 01, 2006 by [tm_jee](#).

If there's a page called home.jsp under the 'agent' namespace, using

```
<ww:url namespace="agent" action="myAction" />
```

would generate

```
http://<host>:<port>/<context>/agent/agent/myAction.action
```

WebWork (DefaultActionMapper) needs namespace to start with slash '/'

```
<ww:url namespace="/agent" action="myAction" />
```

should generate the desired url

```
http://<host>:<port>/<context>/agent/myAction.action
```

Why does WebWork fail to find some javascript

This page last changed on Sep 27, 2006 by [tm_jee](#).

It might be because webwork's FilterDispatcher is not being configured. Since WebWork 2.2.2, ServletDispatcher is being deprecated in favour of FilterDispatcher. FilterDispatcher could be configured as follows

```
<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>
```

It by default provides all static resources having uri of '/webwork/...'

Why does WebWork failed when trying to validate my xml configuration file against its dtd

This page last changed on Feb 19, 2007 by [tm_jee](#).

One possibility would be that due to Proxy/firewall. To configured the proxy, one could have

```
%JDK_HOME%/jre/lib and placed the properties file called net.properties the following entries:  
http.proxyHost=%proxy_address%  
http.proxyPort=%proxy_port%
```

For more information have a look [here](#)

Thanks Macros.

Why does webwork keeps calling my action's properties (eg. my getModel if my action implements ModelDriven) multiple times

This page last changed on Feb 09, 2006 by [tm_jee](#).

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 does WebWork only manage to retrieve attribute from HttpSession the second time around

This page last changed on Dec 16, 2006 by [tm_jee](#).

WebWork doesn't create http session by default. It might be worth looking into "createSession" interceptor and have it at the top of the interceptor stack where one need to retrive an attribute from http session such that a session will be auto-created.

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

Why does WebWork's Rich Text Editor does not render

This page last changed on Oct 05, 2006 by [tm_jee](#).

It was reported that with

```
toolbarCanCollapse="false"
```

FCKEditor does not render or render itself partially. See [here](#) for more info

Why does WW ignore my message when its enclosed in CDATA

This page last changed on Jul 17, 2006 by [ccajina](#).

Try not to separate the message tag and CDATA portion. The parser might create a text node enclosing the CDATA if it sees any space, and xwork just reads the first node (therefore, an empty text). Example:-

```
<field name="password">
  <field-validator type="requiredstring">
    <message><![CDATA[ Debe proporcionar una contraseÃ±a para el usuario. ]]></message>
  </field-validator>
</field>
```

Note: there's no spacing in the following line

```
<message><![CDATA[ ... ]]></message>
```

Contributor: Ruben

Why doesn't WebWork's If tag evaluate test="

This page last changed on Oct 16, 2006 by [tm_jee](#).

try either

```
<ww:if test="#parameters.search[0] == 'something'">
  ...
</ww:if>
```

or

```
<ww:if test="#parameters.search[0] == 'something'">
  ...
</ww:if>
```

Why to use search [0]?

The reason is that #parameters would return a Map, WebWork conveniently make request parameters into a Map to make unit testing action easier and WebWork also uses

```
request.getParameterMap();
```

which returns a map where the key is the parameter (String) while the value is an array of the parameter value (Array of String), hence the need to use [0].

Why isn't my Prepare interceptor being executed

This page last changed on Mar 30, 2006 by [phil](#).

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 Oct 02, 2006 by [goobsoft](#).

```
<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>
```

Old Syntax

```
<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>
```

The Vault

This page last changed on Feb 19, 2007 by [tm_jee](#).

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 think 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 & 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 from 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 & ;& ; (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 environments, 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 override files like:

[http://localhost:8080/showcase/webwork/jscalendar/lang/calendar-\(your language\).js](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

Datepicker does not allow time editing, or works with wrong date format

- [#48644](http://forums.opensymphony.com/thread.jspa?messageID=48644)

The trick is to format the date correctly before sending it to the datepicker, AND telling the datepicker what format you are using.

```
<ww:date name="#{new java.util.Date()}" format="dd-MM-yyyy hh:mm" id="date"/>
<ww:datepicker value="#{#date}" showstime="#{true}" format="%d-%m-%Y %H:%M"/>
```

Note: of course you should use property keys here, but you'll have to specify different keys, since the formatting expression for java and javascript are different (but should result in the same format !):

```
my.java.datetime.format=dd-MM-yyyy hh:mm
my.javascript.datetime.format=%d-%m-%Y %H:%M
```

And the use the following snippet instead:

```
<ww:date name="%{new java.util.Date()}" format="%{getText('my.java.datetime.format')}" id="date"/>
<ww:datepicker value="#{#date}" showstime="%{true}" format="%{getText('my.javascript.datetime.format')}"/>
```

Having DisplayTag's table living side by side with WebWork's Ajax TabPanel

- <http://forums.opensymphony.com/thread.jspa?messageID=117542>

Token Interceptor doesn't play well with ExecuteAndWait Interceptor

- <http://forums.opensymphony.com/thread.jspa?threadID=37690&tstart=0>

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 security 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

WebSphere doesn't recognize all the properties file (eg. webwork.properties, default.properties) in WebWork Web Application

- <http://forums.opensymphony.com/thread.jspa?threadID=26068>

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>

Examples

Steps on getting image using WW action

- <http://forums.opensymphony.com/thread.jspa?messageID=74472>

h2, Xml Parsing

- <http://forums.opensymphony.com/thread.jspa?messageID=100361>

There seems to be some problem when in appropriate parser is being used, resulting in WebWork not being able to recognize elements xwork.xml properly, typically <result> without <param>

Freemarker

Strange Errors/Warnings when using displaytag with freemarker

- <http://forums.opensymphony.com/thread.jspa?messageID=104429#104429>

Freemarker errors (especially on select tag, typically on websphere) under concurrent access

- <http://forums.opensymphony.com/thread.jspa?messageID=111446#111446>

JSP 2.1 Expression Language And WebWork Tag

This page last changed on Mar 15, 2007 by [phil](#).



Content of this page is to be verified of its validity.

With JSP 2.1 pattern like \${} and #{} are valid expression (immediate and deferred expression respectively).

From JSP 2.1 specification the following is how a container should treat page level expression in a JSP page.

JSP Configuration (el-ignored)	Page Directive (isELIgnored)	EL Encountered
unspecified	unspecified	Ignored if <= 2.3 web.xml is used, evaluate otherwise
false	unspecified	Evaluate
true	unspecified	Ignored
don't care	false	Evaluate
don't care	true	Ignored

The following is how container should treat tag directive should evaluate expression or not.

Tag Directive isELIgnored	EL Encountered
unspecified	Evaluate
false	Evaluate
true	Ignored

Both tables above are found in JSP 2.1 specification at page 80.

The JSP 2.1 expression like #{} has different meaning in OGNL expression, in the later it means create a Map. However, using WebWork tags like

```
<ww:set name="myMap" value="#{'one':'ONE', 'two':'TWO', 'three':'THREE'}" />
```

would still continue to work, if it doesn't for some reason, another alternative would be to replace it with

```
<ww:set name="myMap" value="#{@java.util.HashMap@{'one':'ONE', 'two':'TWO', 'three':'THREE'}}" />
```

if order is important, we could use

```
<ww:set name="myMap"
```

```
"#{@java.util.LinkedHashMap@{ 'one':'ONE', 'two':'TWO', 'three':'THREE' }}" />
```

The first version should continue to work because WebWork tags are defined using tld

```
<taglib>
...
<jsp-version>1.2</jsp-version>
...
</taglib>
```

as oppose to

```
<taglib>
...
<jsp-version>2.1</jsp-version>
...
</taglib>
```

where JSP 2.1 dfferred expression (\${}) will be parsed.

Using WebWork tags like

```
<ww:url id="url" action="myAction" namespace="/namespace" />
<ww:a href="#">#{#url}
```

should contine to work regardless of whether its in a JSP 2.1 container or not, as the syntax recongnized by JSP 2.1 containers are \${} and #\${}

With JSP 2.1 containers we could do

```
<ww:url id="url" action="myAction" namespace="/namespace" />
<ww:set name="myUrl" value="#">#{#url}
```

such that WebWork's tag could live side by side with JSP 2.1 EL

As far as WebWork's tag is concerned, it doesn't have tag directive isELIgnored specified, which means that it will evalute JSP 2.1 EL as indicated in the table above. In fact we could do stuff like

```
<ww:set name="varOne" value="#">{'apple'}
```

```
<ww:set name="varTwo" value="#">{'orange'}
```

```
<ww:property value="#">{#varOne+ and ${requestScope.varTwo}}
```

which will print out

```
apple and orange
```

The explanation being, JSP 2.1 EL will be parsed by the container, such that the WebWork property tag

OGNL value expression

```
%{#varOne+' and ${requestScope.varTwo}'}
```

will now be

```
%{#varOne+ ' and orange'}
```

before it is passed to WebWork's property tag, which upon evaluation will become

```
apple and orange
```

Some other information (web.xml) that might be usefull :-

```
<jsp-property-group>
  <url-pattern>*.jsp</url-pattern>

  <!-- ignore JSP 2.1 expression language -->
  <el-ignored>true</el-ignored>

  <!-- don't allow scriptlets -->
  <scripting-invalid>true</scripting-invalid>

  <!-- allow differed syntax as literal -->
  <deferred-syntax-allowed-as-literal>true</deferred-syntax-allowed-as-literal>
</jsp-property-group>
```

or at page level

```
<%@page isELIgnored="true" scriptingInvalid="true" deferredSyntaxAllowedAsLiteral="true" %>
```

References

- [1] - <http://today.java.net/pub/a/today/2006/03/07/unified-jsp-jsf-expression-language.html>
- [2] - http://java.sun.com/products/jsp/reference/techart/unifiedEL.html#Now_Its_Your_Turn

Misc

This page last changed on Mar 23, 2006 by [plightbo](#).

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 Sep 13, 2005 by matt_dowell.

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 at 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
 - 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 Jun 18, 2004 by [plightbo](#).

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"
"yaxslable=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 its 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.

Cookbook

This page last changed on Mar 23, 2006 by [phil](#).



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 Aug 29, 2005 by [plightbo](#).

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 Nesešek (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 Nov 10, 2005 by [tm_jee](#).

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 doesn't exist, it will search request/session/application maps respectively.

Application, Session, Request objects in jsp

This page last changed on Feb 22, 2007 by [phil](#).

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

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



You no longer have to use the set tag to use objects in JSTL - they are automatically provided in WW 2.2.x.

```
<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 Nov 30, 2004 by [jcarreira](#).

```
$req.getSession().getServletContext().getAttribute(...)  
$req.getSession().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 Dec 10, 2004 by [sutter2k](#).

The follow snippet might be useful during debugging to list the properties inside an arbitrary 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.

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

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

This page last changed on Nov 30, 2004 by [jcarreira](#).

```
<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.

File Upload Interceptor

This page last changed on Jan 07, 2007 by [tm_jeet](#).

✖ Jakarta Commons Fileupload Dependencies

When using Jakarta's Commons Fileupload, eg.

```
webwork.multipart.parser=jakarta
```

The following libraries are required

- jakarta commons io
- jakarta commons codec
- jakarta commons fileupload
- jakarta commons logging

✖ Limiting Upload File size using WebWork and Jakarta Commons Fileupload

When using WebWork and Jakarta's Common Fileupload with intend to limit upload file size, do take note that the following entry will be ignored

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

Instead use a the maximumSize property in FileUploadInterceptor eg.

```
<action ...>
    <interceptor-ref name="fileUpload">
        <param name="maximumSize" value="..."/>
    </interceptor-ref>
    ...
</action>
```

Interceptor that is based off of MultiPartRequestWrapper, which is automatically applied for any request that includes a file. It adds the following parameters, where [File Name] is the name given to the file uploaded by the HTML form:

- [File Name] : File - the actual File
- [File Name]ContentType : String - the content type of the file
- [File Name]FileName : String - the actual name of the file uploaded (not the HTML name)

You can get access to these files by merely providing setters in your action that correspond to any of the three patterns above, such as setDocument(File document), setDocumentContentType(String contentType), etc.

See the example code section.

This interceptor will add several field errors, assuming that the action implements ValidationAware. These error messages are based on several i18n values stored in webwork-messages.properties, a default i18n

file processed for all i18n requests. You can override the text of these messages by providing text for the following keys:

- webwork.messages.error.uploading - a general error that occurs when the file could not be uploaded
- webwork.messages.error.file.too.large - occurs when the uploaded file is too large
- webwork.messages.error.content.type.not.allowed - occurs when the uploaded file does not match the expected content types specified

Parameters

- maximumSize (optional) - the maximum size (in bytes) that the interceptor will allow a file reference to be set on the action. Note, this is **not** related to the various properties found in webwork.properties. Default to approximately 2MB.
- allowedTypes (optional) - a comma separated list of content types (ie: text/html) that the interceptor will allow a file reference to be set on the action. If none is specified allow all types to be uploaded.

Extending the Interceptor

You can extend this interceptor and override the `#acceptFile` method to provide more control over which files are supported and which are not.

Examples

```
<action name="doUpload" class="com.examples.UploadAction">
    <interceptor-ref name="fileUpload"/>
    <interceptor-ref name="basicStack"/>
    <result name="success">good_result.ftl</result>
</action>
</pre>
```

And then you need to set encoding `multipart/form-data` in the form where the user selects the file to upload.

```
<pre>
<ww:form action="doUpload" method="post" enctype="multipart/form-data">
    <ww:file name="upload" label="File"/>
    <ww:submit/>
</ww:form>
</pre>
```

And then in your action code you'll have access to the File object if you provide setters according to the naming convention documented in the start.

```
<pre>
public com.examples.UploadAction implements Action {
    private File file;
    private String contentType;
    private String filename;

    public void setUpload(File file) {
        this.file = file;
    }
}</pre>
```

```
public void setUploadContentType(String contentType) {
    this.contentType = contentType;
}

public void setUploadFileName(String filename) {
    this.filename = filename;
}

...
}

</pre>
```

Setting parameters example:

```
<interceptor-ref name="fileUpload">
    <param name="allowedTypes">
        image/png,image/gif,image/jpeg
    </param>
</interceptor-ref>
```

GroovyResult

This page last changed on Jun 02, 2005 by [phil](#).

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 !

Handling IoC Components to Interceptors and Validators

This page last changed on May 15, 2005 by [dhardiker](#).

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 initializing 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 reinstantiated - this will mean that you may experience 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 Nov 12, 2005 by [tm_jee](#).

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 Jan 18, 2007 by [rgielen](#).



Date formatting can be done using the [date](#) tag.

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 -->
<ww:text name="cart.items"><ww:param name="value" value="cartItems"/></ww:text>
```

The above tag will work as follows: the passing of the `value` parameter 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.

```
<ww:text name="last.visit"><ww:param name="userName" /><ww:param
name="getLastVisit(userName)" /></ww:text>
```

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:

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

In WebWork versions before 2.1.7, you would have used the now deprecated - due to missing flexibility - value0 syntax for passing params:

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

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 Sep 23, 2005 by [jhouse](#).

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]</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 Jan 03, 2007 by [phil](#).

Order of interceptors:

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

```
<interceptor-stack name="myStack">
    <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.



Interceptors 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 Jan 03, 2007 by [phil](#).

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 grab 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 Jan 16, 2006 by [phil](#).

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;
    }
}
```

Explanation

```
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 separation between Action and View can be made cleaner. A chart-agnostic 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 Jan 07, 2006 by [plightbo](#).

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 Dec 13, 2005 by [phil](#).

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
```

```

ServletContext.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.

Setting up Eclipse with Tomcat

This page last changed on Mar 07, 2006 by [icewind0](#).

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 Eclipse 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 declaration 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 you 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 see 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 ready to start Tomcat so click the icon **Start Tomcat**. If everything works you will see Tomcat startup and output in the console panel in Eclipse. Now point your web browser 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 truly 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.

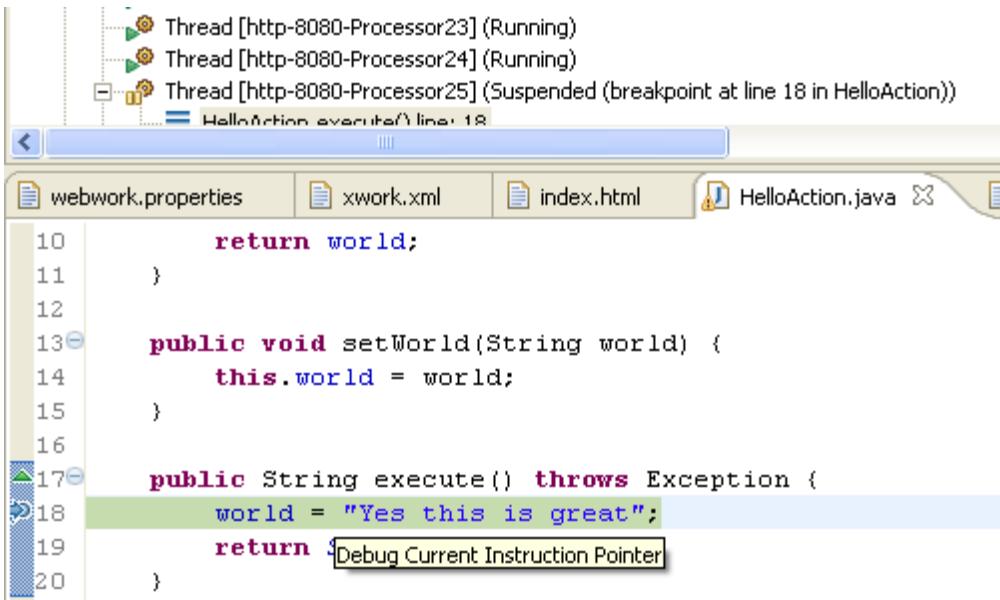
Hot code replace failed

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 easier.



The screenshot shows a Java debugger interface. At the top, there's a tree view of threads: 'Thread [http-8080-Processor23] (Running)', 'Thread [http-8080-Processor24] (Running)', and 'Thread [http-8080-Processor25] (Suspended (breakpoint at line 18 in HelloAction))'. Below the threads is a tab bar with 'HelloAction.java' selected. The code editor shows the following Java code:

```
10     return world;
11 }
12
13 public void setWorld(String world) {
14     this.world = world;
15 }
16
17 public String execute() throws Exception {
18     world = "Yes this is great";
19     return Debug Current Instruction Pointer
20 }
```

The line 'world = "Yes this is great";' is highlighted in green, indicating it is the current instruction pointer. A tooltip 'Debug Current Instruction Pointer' is visible over the return statement.

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 Jan 07, 2006 by [plightbo](#).

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 candidates, 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, quantity 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 Dec 19, 2005 by [plightbo](#).



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 application 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 explicitly 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 Jan 07, 2006 by [plightbo](#).

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 privileges 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 Jun 18, 2004 by [plightbo](#).

```
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 Jun 18, 2004 by [plightbo](#).

```
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; }

/**
 * Sets 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 Jun 18, 2004 by [plightbo](#).

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">

```

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 23, 2006 by [germuska](#).

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-api/1.0/dom3-xml-api-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 Feb 22, 2007 by [phil](#).



JSTL support is fully integrated. You no longer need to use this approach.

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++) {
            if (!ignore.contains(descr[i].getName())) {
                try {
                    props.put(descr[i].getName(), PropertyUtils.getProperty(bean, descr[i].getName()));
                } catch (Exception e) {
                }
            }
        }
        return props;
    }
}
```

```

        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 Feb 22, 2007 by [phil](#).



As of WebWork 2.2, [Spring](#) is the preferred IoC container

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 interceptor, 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 Nov 14, 2005 by [tm_jee](#).

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 Freemaker / 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 Feb 22, 2007 by [phil](#).



Take a look at the [ActionMapper](#) to see how to do this using the `ww:submit` tag and prefixes.

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 similar 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
     */
    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 to 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 Feb 22, 2007 by [phil](#).



WebWork now uses [Themes and Templates](#) instead - you might also want to look at the [SiteMesh](#) integration.

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 Jan 03, 2007 by [phil](#).

File upload using WebWork



This has been replaced by the [fileupload interceptor](#).

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 occurred 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 than 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 occurred 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](#)

Developing WW Ajax Widgets

This page last changed on Oct 08, 2005 by [plightbo](#).

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 occurred">Update</ww:a>
```

This is just a standard WW component. All the attributes defined here must map to a component class that's defined in the taglib.tld. We're not going to go into how the WW components 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 we'll breeze over this.

WW Generated Code

Now let's 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 occurred" showTransportError="true">Update</a>
```

Let's review this file. The dojoType type tells the dojo parser what to do with this tag. This is going to need to correspond 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 BindAnchor... 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 *intercepts* 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 May 21, 2005 by [plightbo](#).

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 Jun 18, 2004 by [plightbo](#).

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](#)).

JavaPolis Opensource Dinner 2006

This page last changed on Nov 22, 2006 by [phil](#).



This is a temporary page - it should be removed after dec. 15, 2006 when the JavaPolis conference is over. Unless something useful can be added here 😊

Introduction

For those of you not familiar with the concept, it's a get-together during the [JavaPolis](#) Conference (13-15 december 2006, Belgium) with lots of influential and less influential open source developers, in combination with great Belgian food and drinks. As usual, there will be dark rituals, world domination plans and lots of opportunities to discuss subjects with fellow Open Source programmers.

The 'open source dinner' is being organized by [Mathias Bogaert](#) for the fourth year in a row, and has become a tradition amongst the open source programmers from various open source projects (including opensymphony, apache, codehaus, spring, ..). It's open for everyone who either is an open source programmer (that is, contributes to an open source project), is a speaker on the conference or has joined the open source dinners in the years before. But we're pretty flexible - if you're a good guy, like to discuss open source projects and are willing to drink the blood of a freshly slaughtered goat, you're welcome to join us.

How do I sign up ?

Well - it's pretty easy: you either send a message to [Mathias Bogaert](#) or Philip Luppens (that's with a dot in between, at gmail), or you can reply to this [thread](#).

It's always possible to sign up during the conference day as well, just use some networking to get in touch (great way to start a conversation), and as long as there are seats available, you're welcome. But we prefer getting notified first - it allows us to inform the restaurant owner (more beer ! more wine !).

When ?

If all goes well, it should be on the first evening of the conference (thus Wednesday **Dec. 13, 2006**).

Where ?

Despite the excellent food and location from last year, we had to look out for a new location that would allow more people to join us (55).



The location Mathias chose is [De Peerdestal](#) in the **Wijngaardstraat 8**, [very close to the location from last year](#) (a 2 minute walk from the 'Grote Markt', the center of Antwerp). Specialities include horse steak, fish, sea fruits and lobster.



The food should be of the same high quality, they should have an even bigger amount of Belgian beer and wine, and prizes should be somewhere around 25 - 40 EUR. (that's about 32 - 52 dollar, they accept Visa/Mastercard/Amex/Diners).

How to get there ?

Well, it's always possible to use a cab/taxi (busses are a possibility too). It's probably possible to catch a ride with some of the belgian/dutch/german attendees (just do the secret handshake, ya know?). But be aware you will definitely need a cab to get back to your hotel (don't worry, plenty of cabs available). Perhaps we can get a bus or some big taxi's to pick us up - but none of that is sure yet. Make sure to check this page for news.

Or, perhaps the good folks from [Atlassian](#) (whom will probably join us) are willing to provide us with another means of transportation ? 😊

Final warnings

Don't forget: Belgian beer (esp. the 'better ones') is very strong. Do not try to 'show off', or by the end of the night, you'll have to crawl home, or you'll wake up somewhere on the deserted shores of a Scottish island, naked, with a tattoo saying 'I love Spring/Duvel/Cameron/..').

Also note that you're not a blocking thread; the conference will continue without you, so don't be late the next day 😊

Cab companies phone numbers:

Metropole Taxi: +32 3 231 3131

Antwerp Tax: +32 3 238 3838

Kroontax: +32 3 646 8383

Learn WW by example

This page last changed on Mar 13, 2005 by [plightbo](#).

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:

1. Readme (this is a work in progress; symbols)
2. Introduction (target audience; Essential, Simple, Intermediate, and Advanced; examples *must* work)
3. Set up WW
4. Essential WW by example
 - a. Actions
5. Simple WW by example: html forms
 - a. Html forms with WW tags, actions, and validation
6. Intermediate WW by example: html forms and pages
 - a. SiteMesh by example
 - b. Struts tag libraries by example
 - c. Interceptors and Validators by example
7. Advanced WW: if you don't test, it isn't for you
 - a. What is IoC?
 - b. JUnit by example
 - c. Mockobjects by example
 - d. Logging by example
 - e. Tomcat commons db connection pool by example
 - f. Hibernate by example
8. Contributing to this wiki guide
 - a. How this guide is different from others
 - b. Rules and conventions

Todo

1. how fit in both jsp and velocity in examples/toc?

Multiple Submit Buttons

This page last changed on Dec 01, 2006 by [franz see](#).

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="Cancel" name="cancel">
```

Action with boolean properties

```
class MyAction extends ActionSupport {
    private boolean submitbutton;
    private boolean cancelbutton;
    public void setSubmitbutton( boolean submitbutton )      {
        this.submitbutton = submitbutton;
    }
    public void setCancelbutton( boolean cancelbutton )     {
        this.cancelbutton = cancelbutton;
    }
    public String execute()      {
        if ( submitbutton )          {
            doSubmit();
            return "submitResult";
        }
        if ( cancelbutton )         {
            doClear();
            return "cancelResult";
        }
        return super.execute();
    }
}
```

Explanation

The boolean properties 'submit' and 'cancel' 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="submitbutton">
<button type="submit" value="Cancel" name="cancelbutton">
```

Action with String properties

```
class MyAction extends ActionSupport {
    private String submitbutton;    private String cancelbutton;

    public void setSubmitbutton( String submitbutton )          {
        this.submitbutton = submitbutton;
    }    public void setCancelbutton( String cancelbutton )      {
        this.cancelbutton= cancelbutton;
    }
    public String execute()      {
        if ( "Submit".equals( submitbutton ) )           {
            doSubmit();
            return "submitResult";
        }
        if ( "Cancel".equals( cancelbutton ) )           {
            doClear();
            return "cancelResult";
        }
        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.)

Warning

- Field names are all in small caps because xwork seems to fail and search the web page otherwise. Values get set to null when not in small caps
- Field names of the buttons must be different from each other. Although having one name for all your submit buttons work in Firefox, it does not with Internet Explorer (IE). The difference is that

Firefox returns only the value of the submit button pressed, while Internet Explorer would return the values of all elements with the given name.

Conclusion

There are other ways to achieve the same functionality. There are pros and cons to each methods.
Feedback welcome.

Performance Tuning

This page last changed on Jan 18, 2007 by [tschneider22](#).

Performance tuning

The following are tips and tricks to squeeze the most performance out of Webwork.

Turn off logging and devMode

DevMode allows reloading of configuration and validation related files, but because they happen on each request, this setting will totally kill your performance.

When using logging, make sure to turn off logging (esp. Freemarker generates a LOT of logging), and check if a level is enabled before printing it, or you will get the cost of the String parsing(concatination anyways).

Do not use interceptors you do not need

If you do not require a full stack of interceptors for an Action, then try using a different one (basicStack), or remove interceptors you do not need.

Use the correct http headers (Cache-Control & Expires)

When returning HTML views, make sure to add the correct headers so browsers knows how to cache them.

Copy the static content from the webwork jar when using the Ajax theme (Dojo) or the Calendar tag

WebWork uses some external javascript libraries and cascading stylesheets for certain themes and tags. These by default are located inside the webwork jar, and a special filter returns them when requesting a special path (/webwork).

Copy the .css and .js files to another directory in your WEB_APP root or even better: use a seperate server for this (lighttpd, apache, ...).

Create a freemarker.properties file in your WEB-INF/classes directory

Create the freemarker.properties file and add the following setting (or whatever value you deem fitting):

```
template_update_delay=60000
```

This value determines how often freemarker checks if it needs to reloads the templates from disk. The default value is 500 ms. Since there is no reason to check if a template needs reloading, it is best to set

this to a very large value. Note that this value is in seconds and freemarker will convert this value to milliseconds.

See also: [Freemarker configuration properties](#)

Copy the /template directory from the webwork jar in your WEB_APP root.

Freemarker fails to properly cache templates when they are retrieved from the classpath. Copying them to the WEB_APP root allows Freemarker to cache them correctly. Freemarker looks at the last modified time of the template to determine if it needs to reload the templates. Resources retrieved from the classpath have no last modified time, so Freemarker will reload them on every request.

When overriding a theme, copy all necessary templates to the theme directory

There's a performance cost when a template cannot be found in the current directory. The reason for this is that Webwork must check for a template in the current theme first before falling back to the parent theme. In the future, this penalty could be eliminated by implementing a missing template cache in Webwork. (Patches would be welcome)

Do not create sessions unless you need them

WebWork does not create sessions unless you ask it to. Note that when you use SiteMesh however, a session will always be created. (See <http://forums.opensymphony.com/thread.jspa?messageID=5688> for details)

When using FreemarkerResult, try to use the Freemarker equivalent rather than using the JSP tags

Freemarker has support for iterating lists, displaying properties, including files, macro's, etc ...

Tabular inputs with HashMap

This page last changed on Jun 17, 2004 by [plightbo](#).

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.

Portlet Support

This page last changed on Jan 03, 2007 by [phil](#).

Portlet Configuration

This page last changed on Jan 03, 2007 by [phil](#).

XWork configuration

The base package of your application should extend the *webwork-portlet-default* package, e.g:

```
<include file="webwork-portlet-default.xml" />  
<package name="view" extends="webwork-portlet-default" namespace="/view">
```



If you're using version 2.2.1 of WebWork, include the file *webwork-default.xml* instead.

Portlet init parameters

Below is the init-param elements that you can set up in *portlet.xml* for configuring the portlet mode -> xwork namespace mappings for the portlet. Basically, you can think of the different portlet modes as different sub-applications, so it can be useful to set up the xwork.xml configuration with different namespaces for the different portlets and modes:

Key	Description	Default
portletNamespace	The namespace for the portlet in the xwork configuration. This namespace is prepended to all action lookups, and makes it possible to host multiple portlets in the same portlet application. If this parameter is set, the complete namespace will be <i>/portletNamespace/modeNamespace/actionName</i>	The default namespace.
viewNamespace	The namespace in the xwork config for the view portlet mode.	The default namespace.
editNamespace	The namespace in the xwork config for the edit portlet mode.	The default namespace.
helpNamespace	The namespace in the xwork config for the help portlet mode.	The default namespace.
defaultViewAction	Name of the action to use as default for the view portlet mode, when no action name is present.	default

defaultEditAction	Name of the action to use as default for the edit portlet mode, when no action name is present.	default
defaultHelpAction	Name of the action to use as default for the help portlet mode, when no action name is present.	default

e.g.

```
<init-param>
    <!-- Portlet namespace -->
    <name>portletNamespace</name>
    <value>/portletA</value>
</init-param>
<init-param>
    <!-- The base namespace of the view portlet mode -->
    <name>viewNamespace</name>
    <value>/view</value>
</init-param>
<init-param>
    <!-- The default action to invoke in view mode -->
    <name>defaultViewAction</name>
    <value>index</value>
</init-param>
```

This snippet from *portlet.xml* will set up the portlet with a namespace of */portletA/*. This means that all requests to this portlet will get the namespace prepended when looking up the action. In addition, the *_view* portlet mode will map to the */view* namespace, so a request for action *myAction* will resolve to the namespace and action */portletA/view/myAction*. It also means that if no action is requested, the default action of *index* will be used for the request.

Portlet phases

The portlet specification describes that a portlet request cycle can consist of two phases, an *event* phase and a *render* phase. In case of an *event* in the portlet, it will always execute before the *render* phase. The *event* phase is typically for changing the state of the application. In a portlet, this is typically when a form is submitted. The *render* phase will then prepare and dispatch to the view. It's recommended that you set up the result from an action that is executed in the *event* phase to point to another action that is executed in the *render* phase, which again is responsible for dispatching to the actual view.

Portlet result dispatching

The *webwork-portlet-default* package defines a special default result type, which is responsible for performing the result logic of an Action execution. Typically, this involves including a JSP for rendering, or preparing a render action if one is configured for the current event action.

This result type has three main modes of execution.

- If the Action is executed in the render phase, it will perform a PortletRequestDispatcher.include(req, res) to the resource specified in the *location* attribute.
- If the Action is executed in the event phase, and the result is an action mapping, it will set a render parameter on the ActionResponse to indicate which Action should be executed in the render phase. This follows good web application design, which promotes the use of a redirect after an event, which in this case means that an Action executed in the event phase will be followed by a redirect to an Action executed in the render phase.
- If the Action is executed in the event phase, and the result is not an action mapping, the result will prepare a special Action called "renderDirect" (specified in the *webwork-portlet-default* package) whose sole purpose is to render the specified web resource (usually a JSP).

The action executed in event mode can pass render parameters to the next action to execute in render mode through a query string in the result configuration:

```
<result name="success">/displayCart.action?userId=${userId}</result>
```

This will set up a *render parameter* called *userId* with the value of the *userId* property of the dispatching action.

Reference

This page last changed on Jan 03, 2007 by [phil](#).

- [Action Chaining](#)
- [ActionMapper](#)
- [Architecture](#)
- [Configuration Files](#)
 - [Reloading configuration](#)
 - [web.xml](#)
 - [web.xml 2.1.x compatibility](#)
 - [webwork-default.xml](#)
 - [webwork.properties](#)
 - [xwork.xml](#)
- [Continuations](#)
- [Dependencies](#)
- [Exception Handling](#)
- [FreeMarker](#)
- [Interceptors](#)
 - [Alias Interceptor](#)
 - [Chaining Interceptor](#)
 - [Component Interceptor](#)
 - [Conversion Error Interceptor](#)
 - [Cookie Interceptor](#)
 - [Create Session Interceptor](#)
 - [Debugging Interceptor](#)
 - [Exception Interceptor](#)
 - [Execute and Wait Interceptor](#)
 - [HibernateAndSpringEnabledExecuteAndWaitInterceptor](#)
 - [Flash Interceptor](#)
 - [I18n Interceptor](#)
 - [Logger Interceptor](#)
 - [Model Driven Interceptor](#)
 - [Parameter Filter Interceptor](#)
 - [Parameter Remover Interceptor](#)
 - [Parameters Interceptor](#)
 - [Prepare Interceptor](#)
 - [Scope Interceptor](#)
 - [Servlet Config Interceptor](#)
 - [Session Invalidiation 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](#)
- [OGNL](#)
 - [OGNL Basics](#)
- [PreResultListener](#)
- [Result Types](#)
 - [Chain Result](#)
 - [Dispatcher Result](#)
 - [Flash Result](#)
 - [FreeMarker Result](#)
 - [WebWork Freemarker Support](#)
 - [HttpHeader Result](#)
 - [JasperReports Result](#)
 - [PlainText Result](#)
 - [Redirect Action Result](#)
 - [Redirect Result](#)
 - [Stream Result](#)
 - [Velocity Result](#)
 - [XSL Result](#)
- [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](#)
 - [fielderror](#)
 - [file](#)
 - [form](#)
 - [head](#)
 - [hidden](#)
 - [label](#)
 - [optgroup](#)
 - [optiontransferselect](#)
 - [password](#)
 - [radio](#)
 - [reset](#)
 - [richtexteditor](#)
 - [select](#)
 - [submit](#)
 - [textarea](#)
 - [textfield](#)
 - [token](#)
 - [updownselect](#)
- [FreeMarker Tags](#)
- [JSP Tags](#)
- [Non Form Tags](#)
 - [a](#)
 - [actionerror](#)
 - [actionmessage](#)
 - [component](#)
 - [date](#)
 - [div](#)
 - [panel](#)
 - [tabbedpane](#)
 - [tabbedPanel](#)
 - [table](#)
 - [tree](#)
 - [treenode](#)
- [Tag Syntax](#)

- [Alt Syntax](#)
- [Themes and Templates](#)
 - [ajax theme](#)
 - [css xhtml theme](#)
 - [Extending Themes](#)
 - [Selecting Themes](#)
 - [simple theme](#)
 - [Template Loading](#)
 - [xhtml theme](#)
- [Velocity Tags](#)
- [Type Conversion](#)
- [Validation](#)
 - [AJAX Validation](#)
 - [Basic Validation](#)
 - [Client Side Validation](#)
 - [AJAX Client Side Validation](#)
 - [Pure JavaScript Client Side Validation](#)
 - [Client Validation](#)
 - [collection validator](#)
 - [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](#)
- [XWork Configuration](#)
 - [Action configuration](#)
 - [Include configuration](#)
 - [Interceptor Configuration](#)
 - [Namespace Configuration](#)
 - [Package Configuration](#)
 - [Result Configuration](#)
 - [Default results](#)
 - [Global results](#)
 - [Views](#)

Action Chaining

This page last changed on Mar 21, 2006 by [phil](#).

Overview

WebWork provides the ability to chain multiple actions into a defined sequence or workflow. This feature works by applying a [Chain Result](#) to a given action, and intercepting its target action's invocation with a [ChainingInterceptor](#).



Warning

In general, Action Chaining is not recommended. However, there are other options, such as the [redirect after post](#) technique.

Chain Result

The [Chain Result](#) is a result type that invokes an action with its own interceptor stack and result. This allows an action to forward requests to a target action, while propagating the state of the source action. Below is an example of how to define this sequence.

```
<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>
```

Another action in the same namespace (or the default "" namespace) can be executed after this action (see [Configuration Files](#)). An optional "namespace" parameter may also be added to specify an action in a different namespace.

Chaining Interceptor

If you need to copy the properties from your previous Actions in the chain to the current action, you should apply the [ChainingInterceptor](#). The interceptor will copy the original parameters from the request, and the ValueStack is passed in to the target action. The source action is remembered by the ValueStack, allowing the target action to access the properties of the preceding action(s) using the ValueStack, and also makes these properties available to the final result of the chain, such as the JSP or Velocity page.

One common use of action chaining is to provide lookup lists (like for a dropdown list of states, etc). Since these actions get put on the ValueStack, these properties will be available in the view. This functionality can also be done using the ActionTag to execute an action from the display page. You may also use the [Redirect Action Result](#) to accomplish this.

ActionMapper

This page last changed on Mar 21, 2006 by [phil](#).

ActionMapper

The ActionMapper is responsible for providing a mapping between HTTP requests and action invocation requests and vice-versa. When given an HttpServletRequest, the ActionMapper may return null if no action invocation request maps, or it may return an ActionMapping that describes an action invocation that WebWork should attempt to try. The ActionMapper is not required to guarantee that the ActionMapping returned be a real action or otherwise ensure a valid request. This means that most ActionMappers do not need to consult WebWork's configuration to determine if a request should be mapped.

Just as requests can be mapped from HTTP to an action invocation, the opposite is true as well. However, because HTTP requests (when shown in HTTP responses) must be in String form, a String is returned rather than an actual request object.

DefaultActionMapper

By default, the DefaultActionMapper is used:

Default action mapper implementation, using the standard *.[ext] (where ext usually "action") pattern. The extension is looked up from the WebWork configuration key **webwork.action.execution**.

To help with dealing with buttons and other related requirements, this mapper (and other ActionMappers, we hope) has the ability to name a button with some predefined prefix and have that button name alter the execution behaviour. The four prefixes are:

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

In addition to these four prefixes, this mapper also understands the action naming pattern of *foo!bar* in either the extension form (eg: *foo!bar.action*) or in the prefix form (eg: *action:foo!bar*). This syntax tells this mapper to map to the action named *foo* and the method *bar*.

Method prefix

With method-prefix, instead of calling baz action's execute() method (by default if it isn't overridden in xwork.xml to be something else), the baz action's anotherMethod() will be called. A very elegant way determine which button is clicked. Alternatively, one would have submit button set a particular value on the action when clicked, and the execute() method decides on what to do with the setted value depending on which button is clicked.

```

<ww:form action="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 prefix

With action-prefix, instead of executing baz action's execute() method (by default if it isn't overridden in xwork.xml to be something else), the anotherAction action's execute() method (assuming again if it isn't overridden with something else in xwork.xml) will be executed.

```

<ww:form action="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 prefix

With redirect-prefix, instead of executing baz action's execute() method (by default it isn't overridden in xwork.xml to be something else), it will get redirected to, in this case to www.google.com. Internally it uses ServletRedirectResult to do the task.

```

<ww:form action="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

With redirect-action-prefix, instead of executing baz action's execute() method (by default it isn't overridden in xwork.xml to be something else), it will get redirected to, in this case 'dashboard.action'. Internally it uses ServletRedirectResult to do the task and read off the extension from the webwork.properties.

```

<ww:form action="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

You can define your own ActionMapper by configuring the ActionMapperFactory:

Factory that creates ActionMappers. This factory looks up the class name of the ActionMapper from

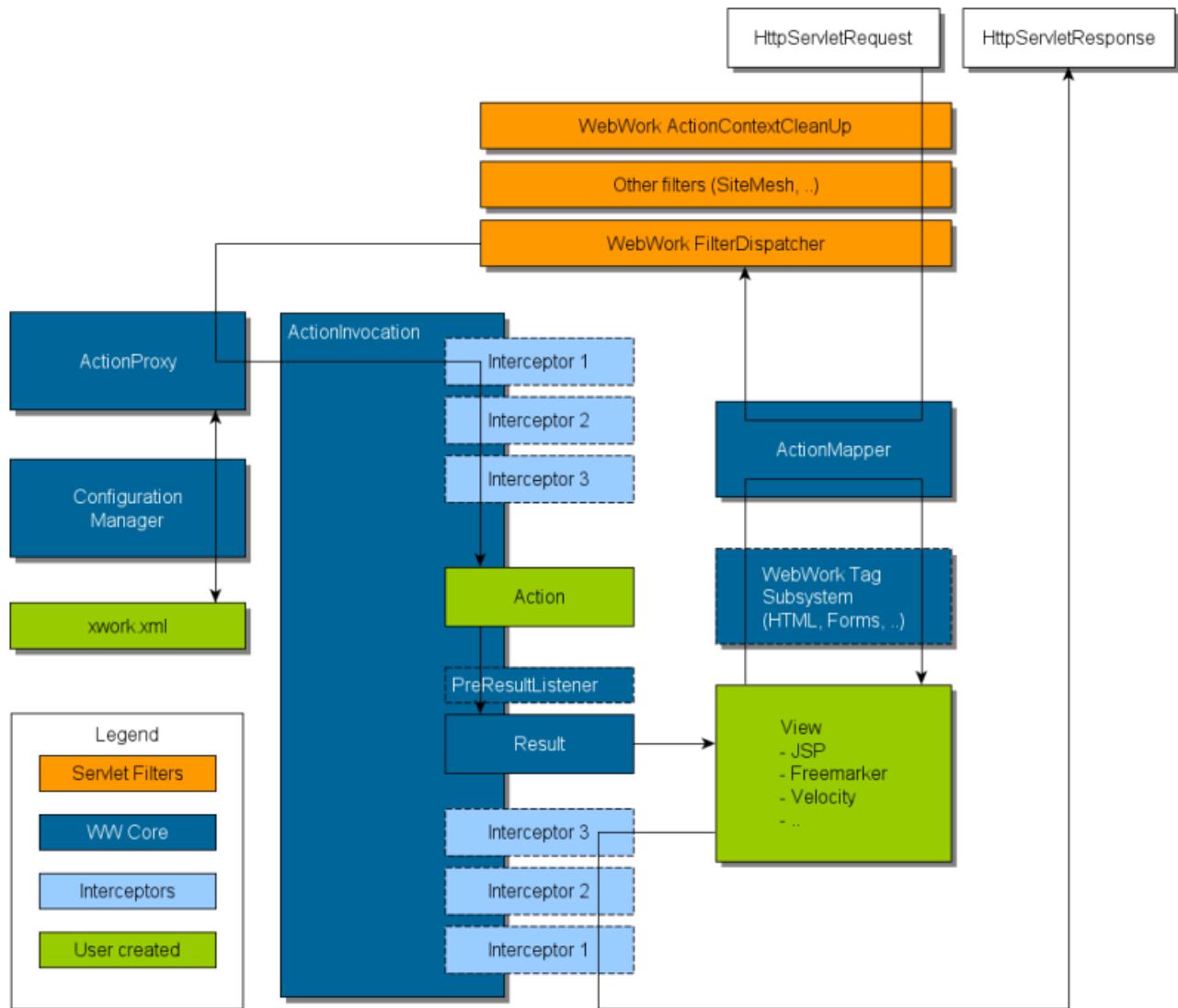
WebWork's configuration using the key **webwork.mapper.class**.

Possible uses of the ActionMapper include defining your own, cleaner namespaces, such as URLs like **/person/1**, which would be similar to a request to **/getPerson.action?personID=1** using the DefaultActionMapper.

Architecture

This page last changed on Dec 07, 2006 by [phil](#).

The WebWork architecture can best be explained with a diagram:



In the diagram, an initial request goes to the Servlet container (such as Tomcat or Resin), the request goes through the standard filter chain. This includes the (optional) **ActionContextCleanUp** filter, which is required if you wish to integrate with technologies such as [SiteMesh](#). Next, the required **FilterDispatcher** is called, which in turn consults the [ActionMapper](#) to determine if the request should invoke an action.

If the ActionMapper determines that an action should be invoked, the FilterDispatcher then delegates to the **ActionProxy**, which in turn consults the WebWork [Configuration Files](#) manager, which finally reads your [xwork.xml](#) file. Next, the ActionProxy creates an **ActionInvocation**, which is responsible for the command pattern implementation. This includes invoking any **interceptors** (the `before()` method) before finally invoking the **action** itself.

Once the action returns, the ActionInvocation is responsible for looking up the proper **result** associated with the **action result code** mapped in xwork.xml. The result is then executed, which often (but not always, as is the case for [Action Chaining](#)) involves a template written in [JSP](#) or [FreeMarker](#) to be rendered. While rendering, the templates can utilize the [Tags](#) provided by WebWork. Some of those components will work with the ActionMapper to render proper URLs for additional requests.



All objects in this architecture (action, result, interceptor, etc) are created by an ObjectFactory. This ObjectFactory is pluggable and is how frameworks like [Spring](#) and [Pico](#) integrate. You can also provide your own ObjectFactory for any reason that requires knowing when objects in WebWork are created.

Finally, the interceptors are executed again (in reverse order, calling the `after()` method) and finally returning back through the filters configured in web.xml. If the ActionContextCleanUp filter is present, the FilterDispatcher will *not* clean up the ThreadLocal **ActionContext**. If the ActionContextCleanUp filter is not present, the FilterDispatcher will cleanup all ThreadLocals.

Configuration Files

This page last changed on Mar 21, 2006 by [phil](#).

Main Configuration Files

WebWork has two main configuration files you need to be aware of: web.xml and xwork.xml. Here you will find out all the information you need for both WebWork's required and optional configuration files.

Below are all the files that you may need to be aware of. Some of this configuration files can be reloaded dynamically, making development much easier. See [Reloading configuration](#) for more information.

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
webwork.properties	yes	/WEB-INF/classes/	WebWork properties
webwork-default.xml	yes	/WEB-INF/lib/webwork-x.x	Default configuration that should be included in xwork.xml
velocity.properties	yes	/WEB-INF/classes/	Override the default velocity configuration

Static Content

Common static content that is needed by webwork (JavaScript and CSS files, etc.) is served automatically by the FilterDispatcher filter. Any request starting with "/webwork/" denotes that static content is required, and then mapping the value after "/webwork/" to common packages in WebWork and, optionally in your class path.

By default, the following packages are searched:

- com.opensymphony.webwork.static
- template

Additional packages can be specified by providing a comma separated list to the configuration parameter named "packages" (configured in web.xml for the FilterDispatcher filter). When specifying additional static content, you should be careful not to expose sensitive configuration information (i.e. database password).

Reloading configuration

This page last changed on May 14, 2004 by [mgreer](#).

Webwork allows for dynamic reloading of xml configuration file (ie, reloading actions.xml).

This allows you to reconfigure your action mapping during development. There may be a slight performance penalty, so this is not recommended for production use.

In order to enable this feature, add the following to your webwork.properties file:

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

web.xml

This page last changed on Feb 24, 2006 by [phil](#).

For those using all the latest features of WebWork and have no requirement for backwards compatibility, configuring web.xml is a matter of adding a single filter and, if you're using JSP, a taglib. However, those upgrading from version 2.1.7 of earlier may need to do a bit more work to get everything in order. See [web.xml 2.1.x compatibility](#) for more information.

The filter is configured as:

An error occurred:

<http://svn.opensymphony.com/svn/webwork/trunk/webwork/webapps/starter/src/webapp/WEB-INF/web.xml?content-type=application/xml&pathrev=1033>
The system administrator has been notified.

For those using JSP, you may also configuration the tag library as:

An error occurred:

<http://svn.opensymphony.com/svn/webwork/trunk/webwork/webapps/starter/src/webapp/WEB-INF/web.xml?content-type=application/xml&pathrev=1033>
The system administrator has been notified.



In case you want to use [SiteMesh](#) for page decoration, you should add some [additional filters](#).

web.xml 2.1.x compatibility

This page last changed on Dec 12, 2005 by bruce@jivesoftware.com.

Before WebWork 2.2, a ServletDispatcher was used to handle action requests. In addition, JSP tags were emulated from within Velocity. WebWork 2.2 made a key changes in this area: The ServletDispatcher was deprecated and replaced with a FilterDispatcher. This generally works perfectly for users who follow the best practices of WebWork, which is what version 2.2 is pushing. However, due to some small behavioral changes in WebWork 2.2, older applications may require the ServletDispatcher.

The biggest change to note is that any application that was including another action, either via a result dispatcher or jsp/ww:include tag, no longer works with the FilterDispatcher. This is because Servlet containers don't support RequestDispatchers out to filter mappings – only servlet mappings are supported. To get around this, you can either change your code to use action chaining in lieu of a result dispatcher and the ww:action tag in lieu of a jsp/ww:include.

As a consequence of switching the FilterDispatcher, JSP tag emulation from within Velocity does not work. While this feature was never fully robust and supported, we recognize that many users take advantage of this feature. As of WebWork 2.2, native Velocity tags are supplied and are the only supported tags within WebWork/Velocity integration.

However, we do provide a deprecated way to avoid changing your code. We recommend that when possible you update your code as suggested. In the meantime, you may add the following Servlets to [web.xml](#):

An error occurred:

<http://svn.opensymphony.com/svn/webwork/trunk/webwork/webapps/starter/src/webapp/WEB-INF/web.xml?content-type=application/xml&pathrev=1000>
The system administrator has been notified.

webwork-default.xml

This page last changed on Jan 08, 2006 by [plightbo](#).

A base configuration file named webwork-default.xml is included in the webwork jar file. This file may be included at the top of your xwork.xml file to include the standard configuration settings without having to copy them, like so:

```
<!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>
```

The contents of webwork-default.xml are here:

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

This file defines all of the default bundled results and interceptors and many interceptor stacks which you can use either as-is or as a basis for your own application-specific interceptor stacks. **Notice the name of the package is "webwork-default".**

webwork.properties

This page last changed on Mar 13, 2007 by [tm_jee](#).

WebWork uses a number of properties that can be changed to fit your needs. To change them, specify your values in webwork.properties in the classpath (typically /WEB-INF/classes). The list of properties can be found in default.properties (inside webwork.jar):

```
### Webwork default properties
###(can be overridden by a webwork.properties file in the root of the classpath)
###

### Specifies the Configuration used to configure webwork
### one could extend com.opensymphony.webwork.config.Configuration
### to build one's customize way of getting the configurations parameters into webwork
# webwork.configuration=com.opensymphony.webwork.config.DefaultConfiguration

### This can be used to set your default locale and encoding scheme
# webwork.locale=en_US
webwork.i18n.encoding=UTF-8

### if specified, the default object factory can be overridden here
### Note: short-hand notation is supported in some cases, such as "spring"
### Alternatively, you can provide a com.opensymphony.xwork.ObjectFactory subclass name
here
# webwork.objectFactory = spring

### specifies the autoWiring logic when using the SpringObjectFactory.
### valid values are: name, type, auto, and constructor (name is the default)
webwork.objectFactory.spring.autoWire = name

### indicates to the webwork-spring integration if Class instances should be cached
### this should, until a future Spring release makes it possible, be left as true
### unless you know exactly what you are doing!
### valid values are: true, false (true is the default)
webwork.objectFactory.spring.useClassCache = true

### if specified, the default object type determiner can be overridden here
### Note: short-hand notation is supported in some cases, such as "tiger" or "notiger"
### Alternatively, you can provide a com.opensymphony.xwork.util.ObjectTypeDeterminer
implementation name here
### Note: if you have the xwork-tiger.jar within your classpath, GenericsObjectTypeDeterminer
is used by default
### To disable tiger support use the "notiger" property value here.
#webwork.objectTypeDeterminer = tiger
#webwork.objectTypeDeterminer = notiger

### Parser to handle HTTP POST requests, encoded using the MIME-type multipart/form-data
# webwork.multipart.parser=cos
# webwork.multipart.parser=pell
webwork.multipart.parser=jakarta
# uses javax.servlet.context.tempdir by default
webwork.multipart.saveDir=
webwork.multipart.maxSize=2097152

### Load custom property files (does not override webwork.properties!)
# webwork.custom.properties=application,com/webwork/extension/custom

### How request URLs are mapped to and from actions
webwork.mapper.class=com.opensymphony.webwork.dispatcher.mapper.DefaultActionMapper

### Used by the DefaultActionMapper
### You may provide a comma separated list, e.g. webwork.action.extension=action,jnlp,do
webwork.action.extension=action

### Used by FilterDispatcher
### If true than WW serves static content from inside its jar.
### If false than the static content must be available at <context_path>/webwork
webwork.serve.static=true

### use alternative syntax that requires {} in most places
### to evaluate expressions for String attributes for tags
```

```

webwork.tag.altSyntax=true

### when set to true, WebWork will act much more friendly for developers. This
### includes:
### - webwork.i18n.reload = true
### - webwork.configuration.xml.reload = true
### - raising various debug or ignorable problems to errors
###   For example: normally a request to foo.action?someUnknownField=true should
###                 be ignored (given that any value can come from the web and it
###                 should not be trusted). However, during development, it may be
###   useful to know when these errors are happening and be told of
###   them right away.
webwork.devMode = false

### when set to true, resource bundles will be reloaded on _every_ request.
### this is good during development, but should never be used in production
webwork.i18n.reload=false

### Standard UI theme
### Change this to reflect which path should be used for JSP control tag templates by default
webwork.ui.theme=xhtml
webwork.ui.templateDir=template
#sets the default template type. Either ftl, vm, or jsp
webwork.ui.templateSuffix=ftl

### Configuration reloading
### This will cause the configuration to reload xwork.xml when it is changed
webwork.configuration.xml.reload=false

### Location of velocity.properties file. defaults to velocity.properties
# webwork.velocity.configfile = velocity.properties

### Comma separated list of VelocityContext classnames to chain to the WebWorkVelocityContext
# webwork.velocity.contexts =

### used to build URLs, such as the UrlTag
webwork.url.http.port = 80
webwork.url.https.port = 443

### possible values are: none, get or all
webwork.url.includeParams = get

### Load custom default resource bundles
# webwork.custom.i18n.resources=testmessages,testmessages2

### workaround for some app servers that don't handle HttpServletRequest.getParameterMap()
### often used for WebLogic, Orion, and OC4J
webwork.dispatcher.parametersWorkaround = false

### configure the Freemarker Manager class to be used
### Allows user to plug-in customised Freemarker Manager if necessary
### MUST extends off com.opensymphony.webwork.views.freemarker.FreemarkerManager
#webwork.freemarker.manager.classname=com.opensymphony.webwork.views.freemarker.FreemarkerManager

### See the WebWorkBeanWrapper javadocs for more information
webwork.freemarker.wrapper.altMap=true

### configure the XSLTResult class to use stylesheet caching.
### Set to true for developers and false for production.
webwork.xslt.nocache=false

### insert Freemarker's Sitemesh applydecorator transform to be put
### into freemarker's model allowing sitemesh's applydecorator tag to
### be used in freemarker's page eg.
### <@sitemesh.applydecorator name="someDecorator" page="/pages/somePage.ftl" />
#
webwork.freemarker.sitemesh.applyDecoratorTransform=true

```

Additional information Development Mode (aka devMode)

What does it do ?

- When enabled, WebWork will reload the **resource bundles on every request** (meaning that the resource bundle (.properties files), could be change instantly and have the changes reflected on the next request).



This option can also be configured independently via the webwork.i18n.reload entry}

- It will also **reload the xml configuration files** ([xwork.xml](#)), the **validation files**, etc, on every request. This is useful for testing or finetuning those configurations, without having to redeploy your application every time.



This option can also be configured independently via `webwork.configuration.xml.reload = true`

- And thirdly, perhaps this setting is less widely known, and therefore a source of much confusion: it will **raise the level of debug or normally ignorable problems to errors**. For example: during **submition of a field that cannot be set on an action** hereafter called the 'someUnknownField', it will normally be ignored. However, when in development mode, **an exception will be logged**, indicating that an invalid field was submitted. This is very useful for debugging or testing large forms, but can also be confusing if relying on parameters in your request that are not set on the action, but are used directly in your view layer.



bad practice, input from the web should always be validated).



By default, the development mode is disabled, because it has a significant impact on [performance](#), since the entire configuration will be reloaded on every request.

This page last changed on Dec 20, 2006 by [phil](#).

Example of xwork dtd

```
<!--
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
>

```

Example of xwork.xml

An error occurred:

<http://svn.opensymphony.com/svn/webwork/webapps/showcase/src/webapp/WEB-INF/classes/xwork-person.xml>.
The system administrator has been notified.

Do not forget to include the webwork-default configuration files; they contain pre-made stacks, the result types, validators and more. To include them, add the following line after your xwork root element:

```

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

```

For more information about the configuration details see [XWork Configuration](#)

Continuations

This page last changed on Mar 07, 2007 by [maydeen](#).

Continuations are a feature in WebWork, borrowed from the [RIFE project](#), that allow for extremely simple state management and wizard-like functionality.



Continuations are currently experimental, and as such, we cannot recommend they be used for heavy-use production deployments at this time. We will continue to work with the community to stabilize and enhance this feature until we are confident it can be used in the most extreme site traffic and use-cases.

Setting it Up

Setting up continuation support requires identifying the base package that your classes are in. This is done in [webwork.properties](#) using the key **webwork.continuations.package**. Typically, this can be the root package that your classes are found in, such as **com.acme**.

Once you've done this, WebWork will analyze your classes and automatically apply continuation support for any class that uses the continuation features - specifically a class that extends ActionSupport that has an **execute()** method that calls a **pause()** method.

URL Concerns

Because continuations require the state of your flow be managed by WebWork, it is up to you to make sure your application inform WebWork what the flow's ID is. This is done via a **continue** parameter that provides a unique ID for every request in the flow. Assuming you are generating your URLs using the [URL](#) tag or the [Form](#) tag, this is handled for you automatically. If you are *not* using these tags, continuations will not work.

Interceptor Concerns

Because continuations radically change the way your actions are invoked, it is important to understand how this affects interceptors. The most important think to know is that continuations kick in only when the **execute()** method is called. This means that on every request (regardless of whether it is a new request or a continuation), the interceptors will be called. This is what makes it possible to apply new request parameters to your action even though the rest of the call stack appears to look the same.

This is generally exactly what you would want, except some interceptors, namely the [Execute and Wait Interceptor](#) and possibly the [Token Session Interceptor](#), have very different expectations about the workflow/lifecycle of the action invocation. In these cases, continuations should not be used.

Example

Getting started with continuations is extremely simple. The biggest thing to get used to is the very different conversational style with application workflow. Typically, you might have used session variables or hidden form fields to pass the state around. Using continuations, you use the Java language to handle that state. See the following body of a Guess class extending ActionSupport:

```
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;
    }
}
```

Note how the class keeps the state (tries, in this example) as a local variable in the execute() method. WebWork's continuations will automatically pick up the invocation after the pause() method call and will restore all local variables, as if the logical loop is continuing "magically" (read on for more info on how it works).

The view is nothing special, except for that fact that it adheres to the URL concerns and uses the [Form](#) tag to render the URL. This makes sure that the **continue** parameter is included in all requests.

```
<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

Continuations are not magic, though sometimes they might seem like they are. In fact, they work by using some very intelligent byte-code manipulation. This means that **in order to use continuations, your deployment environment allow for custom class loaders to handle loading your actions.** Typically this is not a problem, but it should be called out.

Once the class is requested to be loaded, WebWork will hand off the request to the RIFE/Continuations module, which will then check a few conditions:

1. Does the class extend ActionSupport?
2. Does the class have an execute() method?
3. In the execute() method, are there any calls to pause()?

If the answer is yes to all three conditions, the class is then instrumented and the execute() method is rewritten with try/catch code, goto statements, and intelligent "state restoration" code. All this happens transparently and does not affect the ability to debug the class or otherwise code it.

See the pause() method JavaDocs in the ActionSupport class for more info:

Stops the action invocation immediately (by throwing a PauseException) and causes the action invocation to return the specified result, such as #SUCCESS, #INPUT, etc.

The next time this action is invoked (and using the same continuation ID), the method will resume immediately after where this method was called, with the entire call stack in the execute method restored.

Note: this method can **only** be called within the #execute() method.

Dependencies

This page last changed on Jan 03, 2007 by [phil](#).

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 portlets
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" href="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].[ext]">
            </url>
        <url name="contegix">
            <ivy
pattern="http://repository.contegix.com/ivyrep/[organisation]/[module]/ivy-[revision].xml"/>
            <artifact
                pattern="http://repository.contegix.com/ivyrep/[organisation]/[module]/[artifact]-[revision].[ext]">
            </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.

Exception Handling

This page last changed on Mar 21, 2006 by [phil](#).

Overview

Exception mappings are a powerful feature for dealing with an Action that throws an exception. The core idea is that exception thrown during the Action method can be caught and mapped to a result, either global or action scoped results. This is especially useful for frameworks, like Hibernate and Acegisecurity, that throw RuntimeExceptions.

As with many other parts of WebWork, an interceptor is needed to activate the exception mapping functionality. Below is a snippet from webwork-default.xml which has exception mapping already activated.

```
...
<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>
...
```

The next step in exception mapping is to actually map exception to specific results. WebWork provides two ways to declare an exception mapping `<exception-mapping>` - globally or for a specific action. The exception mapping element takes two attributes, `exception` and `result`.

When declaring an exception mapping, the interceptor will find the closest class inheritance match between the exception thrown and the exception declared. The interceptor will examine all declared mappings applicable to the Action, both Action specific and global mappings. The result (either global or Action scope) is then used.

This follows the same rules as a result returned from an Action. It first looks for the result in the action, and then if not found, it looks for the result as a global result.

Below is an example of global and Action scoped exception mappings.

```
<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>

```

In the example above, here is what happens based upon each exception:

- A `java.sql.SQLException` will chain to the `sqlExceptionAction`
- A `com.acme.foo.SecurityException` will redirect to `/login.action`
- Any other exception that extends `java.lang.Exception` will execute the FreeMarker result `rootException` for the page `/WEB-INF/views/exception.ftl`

Exception Values on the ValueStack

By default, the `ExceptionMappingInterceptor` adds the following values to the Value Stack:

- `exception` - The exception object itself
- `exceptionStack` - the value from the stack trace

FreeMarker

This page last changed on Jun 18, 2006 by [phil](#).

FreeMarker is a template Java template language that is a great alternative to [JSP](#). FreeMarker is ideal for situations where your action results can possibly be loaded from outside a Servlet container. For example, if you wished to support plugins in your application, you might wish to use FreeMarker so that the plugins could provide the entire action class and view in a single jar that is loaded from the classloader.

For more information on FreeMarker itself, please visit the [FreeMarker website](#).



FreeMarker is very similar to [Velocity](#), as both are template languages that can be used outside of a Servlet container. The WebWork team recommends FreeMarker over Velocity simply because FreeMarker has better error reporting, support for JSP tags, and slightly better features. However, both are good alternatives to JSP.

Getting Started

Getting started with FreeMarker is as simple as ensuring all the [dependencies](#) are included in your project's classpath. This typically requires simply **freemarker.jar**. Other than that, [webwork-default.xml](#) already configures the [FreeMarker Result](#) needed to map your actions to your templates. You may now try out the following **xwork.xml** configuration:

```
<action name="test" class="com.acme.TestAction">
    <result name="success" type="freemarker">test-success.ftl</result>
</action>
```

Then in **test-success.ftl**:

```
<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.

Servlet / JSP Scoped Objects

The following are ways to obtain Application scope attributes, Session scope attributes, Request scope attributes, Request parameters and SAF Context scope parameters:-

Application Scope Attribute

Assuming there's an attribute with name 'myApplicationAttribute' in the Application scope.

```
<if Application.myApplicationAttribute?exists>
    ${Application.myApplicationAttribute}
</if>
```

or

```
<saf.property value="#{application.myApplicationAttribute}" />
```

Session Scope Attribute

Assuming there's an attribute with name 'mySessionAttribute' in the Session scope.

```
<if Session.mySessionAttribute?exists>
    ${Session.mySessionAttribute}
</if>
```

or

```
<saf.property value="#{session.mySessionAttribute}" />
```

Request Scope Attribute

Assuming there's an attribute with name 'myRequestAttribute' in the Request scope.

```
<if Request.myRequestAttribute?exists>
    ${Request.myRequestAttribute}
</if>
```

or

```
<saf.property value="#{request.myRequestAttribute}" />
```

Request Parameter

Assuming there's a request parameter myParameter (eg. <http://host/myApp/myAction.action?myParameter=one>).

```
<if Parameters.myParameter?exists>
  ${Parameters.myParameter}
</if>
```

or

```
<@saf.property value="%{#parameters.myParameter}" />
```

Context parameter

Assuming there's a parameter with the name myContextParam in SAF context.

```
${stack.findValue('#myContextParam')}
```

or

```
<@saf.property value="%{#myContextParam}" />
```

Template Loading

WebWork looks for FreeMarker templates in two locations (in this order):

1. Web application
2. Class path

This ordering makes it ideal for providing templates inside a fully built jar, but allowing for overrides of those templates to be defined in your web application. In fact, this is how you can override the default UI tags and [Form Tags](#) included with WebWork.

In addition, you can specify a location (directory on your file system) through the 'templatePath' or 'TemplatePath' context variable (in your web.xml). If variable is specified, the content of the directory it points to will be searched first.



This variable is currently NOT relative to the root of your webapp. We suggest placing the templates under WEB-INF anyway

Variable Resolution

In FreeMarker, variables are looked up in several different places, in this order:

1. Built-in variables
2. The value stack
3. The action context
4. Request scope
5. Session scope
6. Application scope

Note that the action context is looked up after the value stack. This means that you can reference the variable without the typical preceding has marker (#) like you would have to when using the JSP `ww:property` tag. This is a nice convenience, though be careful because there is a small chance it could trip you up.

```
<@ww.url id="url" value="http://www.yahoo.com"/>
Click <a href="#">here!
```

The built-in variables that WebWork-FreeMarker integration provides are:

Name	Description
stack	The value stack itself, useful for calls like <code> \${stack.findString('ognl expr')}</code>
action	The action most recently executed
response	The <code>HttpServletResponse</code>
res	Same as response
request	The <code>HttpServletRequest</code>
req	Same as request
session	The <code>HttpSession</code>
application	The <code>ServletContext</code>
base	The request's context path

Tag Support

FreeMarker is a great template language because it has complete tag support. See the [FreeMarker Tags](#) documentation for information on how to use the generic [Tags](#) provided by WebWork. In addition to this, you can use any JSP tag, like so:

```
<#assign mytag=JspTaglibs["/WEB-INF/mytag.tld"]>
<@mytag.tagx attribute1="some ${value}" />
```

Where **mytag.tld** is the JSP Tag Library Definition file for your tag library. Note: in order to use this support in FreeMarker, you must enable the `JSPSupportServlet` documented in [web.xml 2.1.x](#)

[compatibility](#).

Tips and Tricks

There are some advanced features that may be useful when building WebWork applications with FreeMarker.

Type Conversion and Locales

FreeMarker has built in support for formatting dates and numbers. The formatting rules are based on the locale associated with the action request, which is by default set in [webwork.properties](#) but can be over-ridden using the [I18n Interceptor](#). This is normally perfect for your needs, but it is important to remember that these formatting rules are handled by FreeMarker and not by WebWork's [Type Conversion](#) support.

If you want WebWork to handle the formatting according to the [Type Conversion](#) you have specified, you shouldn't use the normal \${...} syntax. Instead, you should use the [property](#) tag. The difference is that the property tag is specifically designed to take an [OGNL](#) expression, evaluate it, and then convert it to a String using any [Type Conversion](#) rules you have specified. The normal \${...} syntax will use a FreeMarker expression language, evaluate it, and then convert it to a String using the built in formatting rules. *This difference is subtle but important to understand.*

Extending

Sometimes you may wish to extend the FreeMarker support provided with WebWork. The most common reason for doing this is that you wish to include your own [Tags](#), such as those that you have extended from the built in WebWork [Tags](#).

To do so, write a new class that extends **com.opensymphony.webwork.views.freemarker.FreemarkerManager** and overrides it as needed. Then add the following to [webwork.properties](#):

```
webwork.freemarker.manager.classname = com.yourcompany.YourFreemarkerManager
```

ObjectWrapper Settings

Once you get familiar with FreeMarker, you will find certain *subtleties* with it that may become frustrating. The most common thing you'll likely run into is the BeansWrapper provided by FreeMarker. If you don't know what this is, don't worry. However, if you do, know this:

The WebWorkBeanWrapper extends the default FreeMarker BeansWrapper and provides almost no change in functionality, **except** for how it handles maps. Normally, FreeMarker has two modes of

operation: either support for friendly map built-ins (?keys, ?values, etc) but only support for String keys; OR no special built-in support (ie: ?keys returns the methods on the map instead of the keys) but support for String and non-String keys alike. WebWork provides an alternative implementation that gives us the best of both worlds.

It is possible that this special behavior may be confusing or can cause problems. Therefore, you can set the **webwork.freemarker.wrapper.altMap** property in webwork.properties to false, allowing the normal BeansWrapper logic to take place instead.

Syntax Notes

As of FreeMarker 2.3.4, an alternative syntax is supported. This alternative syntax is great if you find that your IDE (especially IntelliJ IDEA) makes it difficult to work with the default syntax. You can read more about this syntax [here](#).

Interceptors

This page last changed on Jan 09, 2007 by [tm_jee](#).

See [Interceptor Configuration](#) for basic information about how interceptors are configured.

Overview

An interceptor is a stateless class that follows the interceptor pattern, as found in {@link javax.servlet.Filter} and in AOP languages.

Interceptors are objects that dynamically intercept Action invocations. They provide the developer with the opportunity to define code that can be executed before and/or after the execution of an action. They also have the ability to prevent an action from executing. Interceptors provide developers a way to encapsulate common functionality in a re-usable form that can be applied to one or more Actions.

Interceptors **must** be stateless and not assume that a new instance will be created for each request or Action. Interceptors may choose to either short-circuit the ActionInvocation execution and return a return code (such as com.opensymphony.xwork.Action#SUCCESS), or it may choose to do some processing before and/or after delegating the rest of the processing using ActionInvocation#invoke().

Webwork & XWork Interceptors

Interceptor classes are also defined using a key-value pair specified in the xwork configuration file. The names specified below come from specified in [webwork-default.xml](#). If you extend the webwork-default package, then you can use the names below. Otherwise they must be defined in your package with a name-class pair specified in the <interceptors> tag.

Interceptor	Name	Description
Alias Interceptor	alias	Converts similar parameters that may be named differently between requests.
Chaining Interceptor	chain	Makes the previous action's properties available to the current action. Commonly used together with <result type="chain"> (in the previous action).
Component Interceptor	component	Enables and makes the components available to the Actions. Refer to components.xml
Conversion Error Interceptor	conversionError	adds conversion errors from the ActionContext to the Action's field errors

Create Session Interceptor	createSession	create an HttpSession automatically, usefull with certain interceptor (eg. TokenInterceptor) when an HttpSession is required in order to work properly
Execute and Wait Interceptor	execAndWait	an interceptor that executes the action in the background and then sends the user off to an intermediate waiting page.
Exception Interceptor	exception	Maps exceptions to a result.
File Upload Interceptor	fileUpload	an interceptor that adds easy access to file upload support. See the javadoc for more info
I18n Interceptor	i18n	remembers the locale selected for a user's session
Logger Interceptor	logger	Outputs the name of the action
Model Driven Interceptor	model-driven	If the action implements ModelDriven, pushes the getModel() result onto the valuestack.
Parameters Interceptor	params	Sets the request parameters onto the action.
Prepare Interceptor	prepare	If the action implements Preparable, calls its prepare() method.
Scope Interceptor	scope	simple mechanism for storing action state in the session or application scope
Servlet Config Interceptor	servlet-config	Give access to HttpServletRequest and HttpServletResponse (think twice before using this since this ties you to the Servlet api)
Static Parameters Interceptor	static-params	Sets the xwork.xml defined parameters onto the action. These are the <param> tags that are direct children of the <action> tag.
Timer Interceptor	timer	Outputs how long the action (including nested interceptors and view) takes to execute
Token Interceptor	token	Checks for valid token presence in action, prevents duplicate form submission
Token Session Interceptor	token-session	Same as above, but storing the submitted data in session when

		handed an invalid token
Validation Interceptor	validation	Performs validation using the validators defined in {Action}-validation.xml
Workflow Interceptor	workflow	Calls the validate method in your action class. If action errors created then it returns the INPUT view.
Parameter Filter Interceptor	paramFilter	Removes parameters from the list of those available to actions etc.
Parameter Remover Interceptor	paramRemover	Removes parameter from parameter map if their param name and value matches a certain configurable value defined through paramNames and paramValues attribute respectively.
Session Invalidation Interceptor	sessionInvalidation	invalidates the http session, either now or in the next comming request where this interceptor is present in the interceptor stack.
Flash Interceptor	flash	store/retrieve action from http session and push them into stack, such that action (and its information stored) will be available after redirect.
Debugging Interceptor	debugging	Printing out content in value stack
Cookie Interceptor	cookie	Inject cookie with a certain configurable name / value into action

Method Filtering

An abstract `Interceptor` that is applied to selectively according to specified included/excluded method lists.

Setable parameters are as follows:

- `excludeMethods` - methods name to be excluded
- `includeMethods` - methods name to be included

NOTE: If method name are available in both `includeMethods` and `excludeMethods`, it will still be considered as an included method. In short `includeMethods` takes precedence over `excludeMethods`.

Interceptors that extends this capability would be :-

- TokenInterceptor
- TokenSessionStoreInterceptor
- DefaultWorkflowInterceptor
- ValidationInterceptor

Interceptor Parameter Overriding

Interceptor's parameter could be overriden through the following ways :-

Method 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">myValidationExcludeMethod</param>
    </interceptor-ref>
    <interceptor-ref name="workflow">
        <param name="excludeMethods">myWorkflowExcludeMethod</param>
    </interceptor-ref>
</action>
```

Method 2:

```
<action name="myAction" class="myActionClass">
    <interceptor-ref name="defaultStack">
        <param name="validation.excludeMethods">myValidationExcludeMethod</param>
        <param name="workflow.excludeMethods">myWorkflowExcludeMethod</param>
    </interceptor-ref>
</action>
```

In the first method, the whole default stack is copied and the parameter then changed accordingly.

In the second method, the refer to an existing interceptor-stack, namely default-stack in this example, and override the validator and workflow interceptor excludeMethods typically in this case. Note that in the tag, the name attribute contains a dot (.) the word before the dot(.) specifies the interceptor name whose parameter is to be overriden and the word after the dot (.) specifies the parameter itself. Essetially it is as follows :-

```
<interceptor-name>. <parameter-name>
```

Note also that in this case the name attribute is used to indicate an interceptor stack which makes sense as if it is refering to the interceptor itself it would be just using Method 1 describe above.

Order of Interceptor Execution

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

```
<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.

Interceptors 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
```

Alias Interceptor

This page last changed on Oct 18, 2005 by [digi9ten](#).

The aim of this Interceptor is to alias a named parameter to a different named parameter. By acting as the glue between actions sharing similar parameters (but with different names), it can help greatly with action chaining.

Action's alias expressions should be in the form of #{"name1" : "alias1", "name2" : "alias2"}. This means that assuming an action (or something else in the stack) has a value for the expression named *name1* and the action this interceptor is applied to has a setter named *alias1*, *alias1* will be set with the value from *name1*.

Parameters

- `aliasesKey` (optional) - the name of the action parameter to look for the alias map (by default this is *aliases*).

Extending the Interceptor

This interceptor does not have any known extension points.

Examples

```
<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 Oct 24, 2005 by [plightbo](#).

An interceptor that copies all the properties of every object in the value stack to the currently executing object, except for any object that implements Unchainable. A collection of optional *includes* and *excludes* may be provided to control how and which parameters are copied. Only includes or excludes may be specified. Specifying both results in undefined behavior. See the javadocs for {@link OgnlUtil#copy(Object, Object, java.util.Map, java.util.Collection, java.util.Collection)} for more information.

It is important to remember that this interceptor does nothing if there are no objects already on the stack. This means two things: One, you can safely apply it to all your actions without any worry of adverse affects. Two, it is up to you to ensure an object exists in the stack prior to invoking this action. The most typical way this is done is through the use of the **chain** result type, which combines with this interceptor to make up the action chaining feature.

Parameters

- **excludes** (optional) - the list of parameter names to exclude from copying (all others will be included).
- **includes** (optional) - the list of parameter names to include when copying (all others will be excluded).

Extending the Interceptor

There are no known extension points to this interceptor.

Examples

```
<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 Jul 10, 2006 by [plightbo](#).

A simple interceptor that applies the WebWork IOC container ComponentManager against the executing action. Note, WebWork IOC is deprecated and it is highly recommended that you look at alternative solutions, such as Spring.

Parameters

- None

Extending the Interceptor

There are no known extension points to this interceptor.

Examples

```
<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 Oct 25, 2005 by [plightbo](#).

To fully document this interceptor, it is best to look at the JavaDocs for the subclass of the interceptor, `ConversionErrorInterceptor`:

This interceptor adds any error found in the `ActionContext`'s `conversionErrors` map as a field error (provided that the action implements `ValidationAware`). In addition, any field that contains a validation error has its original value saved such that any subsequent requests for that value return the original value rather than the value in the action. This is important because if the value "abc" is submitted and can't be converted to an int, we want to display the original string ("abc") again rather than the int value (likely 0, which would make very little sense to the user).

... as well as the JavaDocs for the interceptor itself, `WebWorkConversionErrorInterceptor`:

This interceptor extends `ConversionErrorInterceptor` but only adds conversion errors from the `ActionContext` to the field errors of the action if the field value is not null, "", or {""} (a size 1 String array with only an empty String). See `ConversionErrorInterceptor` for more information, as well as the Type Conversion documentation.

Parameters

- None

Extending the Interceptor

There are no known extension points for this interceptor.

Examples

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="params"/>
    <interceptor-ref name="conversionError"/>
    <result name="success">good_result.ftl</result>
</action>
```

Cookie Interceptor

This page last changed on Jan 09, 2007 by [tm_jee](#).

The aim of this interceptor is to set values in the stack/action based on cookie name/value of interest.

If an asterik is present in cookiesName parameter, it will be assume that all cookies name are to be injected into webwork's action, even though cookiesName is comma-separated by other values, eg (cookie1,*,cookie2).

If cookiesName is left empty it will assume that no cookie will be injected into WebWork's action.

If an asterik is present in cookiesValue parameter, it will assume that all cookies name irrespective of its value will be injected into WebWork's action so long as the cookie name matches those specified in cookiesName parameter.

If cookiesValue is left empty it will assume that all cookie that match the cookieName parameter will be injected into WebWork's action.

The action could implements CookiesAware in order to have a Map of filtered cookies set into it.

Parameters

- cookiesName (mandatory) - Name of cookies to be injected into the action. If more than one cookie name is desired it could be comma-separated. If all cookies name is desired, it could simply be *, an asterik. When many cookies name are comma-separated either of the cookie that match the name in the comma-separated list will be qualified.
- cookiesValue (mandatory) - Value of cookies that if its name matches cookieName attribute and its value matched this, will be injected into WebWork's action. If more than one cookie name is desired it could be comma-separated. If left empty, it will assume any value would be ok. If more than one value is specified (comma-separated) it will assume a match if either value is matched.

Extending the Interceptor

populateCookieValueIntoStack - this method will decide if this cookie value is qualified to be populated into the value stack (hence into the action itself)
injectIntoCookiesAwareAction - this method will inject selected cookies (as a java.util.Map) into action that implements CookiesAware.

Examples

```
<!--
This example will inject cookies named either 'cookie1' or 'cookie2' whose
value could be either 'cookie1value' or 'cookie2value' into WebWork's action.
-->
<action ... >
<interceptor-ref name="cookie">
```

```

<param name="cookiesName">cookie1, cookie2</param>
<param name="cookiesValue">cookie1value, cookie2value</param>
</interceptor-ref>
...
</action>

<!--
      This example will inject cookies named either 'cookie1' or 'cookie2'
      regardless of their value into WebWork's action.
-->
<action ... >
<interceptor-ref name="cookie">
<param name="cookiesName">cookie1, cookie2</param>
<param name="cookiesValue">*</param>
</interceptor-ref>
...
</action>

<!--
      This example will inject cookies named either 'cookie1' with value
      'cookie1value' or 'cookie2' with value 'cookie2value' into WebWork's
      action.
-->
<action ... >
<interceptor-ref name="cookie">
<param name="cookiesName">cookie1</param>
<param name="cookiesValue">cookie1value</param>
</interceptor-ref>
<interceptor-ref name="cookie">
<param name="cookiesName">cookie2</param>
<param name="cookiesValue">cookie2value</param>
</interceptor-ref>
...
</action>

<!--
      This example will inject any cookies regardless of its value into
      WebWork's action.
-->
<action ... >
<interceptor-ref name="cookie">
<param name="cookiesName">*</param>
<param name="cookiesValue">*</param>
</interceptor-ref>
...
</action>

```

Create Session Interceptor

This page last changed on Feb 25, 2006 by [tm_jee](#).

This interceptor creates the HttpSession. This is particular usefull when using the <@ww.tokten> tag in freemarker templates. The tag **do** require that a HttpSession is already created since freemarker commits the response to the client immediately.

Parameters

- None

Extending the Interceptor

- none

Examples

```
<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>
```

Debugging Interceptor

This page last changed on Jan 09, 2007 by [tm_jee](#).

Provides several different debugging screens to provide insight into the data behind the page. The value of the 'debug' request parameter determines the screen. This interceptor only is activated when devMode is enabled in webwork.properties. The 'debug' parameter is removed from the parameter list before the action is executed. All operations occur before the natural Result has a chance to execute.



The debug interceptor will place the Ognl stack into the session. This can cause problems when the session is attempted to be serialised (such as when using a session based clustering service, when Tomcat startsup/shutdown etc.) as your stack may contain non-serializable objects such as services or loggers (in particular log4j Logger which does not implement serializable).

Parameters

The value of the 'debug' request parameter determines the screen.

- `xml` - Dumps the parameters, context, session, and value stack as an XML document.
- `console` - Shows a popup 'OGNL Console' that allows the user to test OGNL expressions against the value stack. The XML data from the 'xml' mode is inserted at the top of the page.
- `command` - Tests an OGNL expression and returns the string result. Only used by the OGNL console.

Extending the Interceptor

There's no intended extension points

Examples

```
<action ...>
  <interceptor-ref name="debugging" />
  ...
</action>
```

Exception Interceptor

This page last changed on Oct 18, 2005 by [digi9ten](#).

This interceptor forms the core functionality of the exception handling feature. Exception handling allows you to map an exception to a result code, just as if the action returned a result code instead of throwing an unexpected exception. When an exception is encountered, it is wrapped with an `ExceptionHolder` and pushed on the stack, providing easy access to the exception from within your result.

Note: While you can configure exception mapping in your configuration file at any point, the configuration will not have any effect if this interceptor is not in the interceptor stack for your actions. It is recommended that you make this interceptor the first interceptor on the stack, ensuring that it has full access to catch any exception, even those caused by other interceptors.

Parameters

- `logEnabled` (optional) - Should exceptions also be logged? (boolean true|false)
- `logLevel` (optional) - what log level should we use (`trace`, `debug`, `info`, `warn`, `error`, `fatal`)? - default is `debug`
- `logCategory` (optional) - If provided we would use this category (eg. `com.mycompany.app`). Default is to use `com.opensymphony.xwork.interceptor.ExceptionMappingInterceptor`.

The parameters above enables us to log all thrown exceptions with stacktrace in our own logfile, and present a friendly webpage (with no stacktrace) to the end user.

Extending the Interceptor

If you want to add custom handling for publishing the Exception, you may override {@link #publishException(com.opensymphony.xwork.ActionInvocation, ExceptionHolder)}. The default implementation pushes the given `ExceptionHolder` on value stack. A custom implementation could add additional logging etc.

Examples

```
<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"/>
        </action>
    </package>
</xwork>
```

```
<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 Oct 18, 2005 by [plightbo](#).

The ExecuteAndWaitInterceptor is great for running long-lived actions in the background while showing the user a nice progress meter. This also prevents the HTTP request from timing out when the action takes more than 5 or 10 minutes.

Using this interceptor is pretty straight forward. Assuming that you are including webwork-default.xml, this interceptor is already configured but is not part of any of the default stacks. Because of the nature of this interceptor, it must be the **last** interceptor in the stack.

This interceptor works on a per-session basis. That means that the same action name (myLongRunningAction, in the above example) cannot be run more than once at a time in a given session. On the initial request or any subsequent requests (before the action has completed), the **wait** result will be returned. **The wait result is responsible for issuing a subsequent request back to the action, giving the effect of a self-updating progress meter.**

If no "wait" result is found, WebWork will automatically generate a wait result on the fly. This result is written in FreeMarker and cannot run unless FreeMarker is installed. If you don't wish to deploy with FreeMarker, you must provide your own wait result. This is generally a good thing to do anyway, as the default wait page is very plain.

Whenever the wait result is returned, the **action that is currently running in the background will be placed on top of the stack**. This allows you to display progress data, such as a count, in the wait page. By making the wait page automatically reload the request to the action (which will be short-circuited by the interceptor), you can give the appearance of an automatic progress meter.

This interceptor also supports using an initial wait delay. An initial delay is a time in milliseconds we let the server wait before the wait page is shown to the user. During the wait this interceptor will wake every 100 millis to check if the background process is done premature, thus if the job for some reason doesn't take to long the wait page is not shown to the user.

This is useful for e.g. search actions that have a wide span of execution time. Using a delay time of 2000 millis we ensure the user is presented fast search results immediately and for the slow results a wait page is used.

Important: Because the action will be running in a separate thread, you can't use ActionContext because it is a ThreadLocal. This means if you need to access, for example, session data, you need to implement SessionAware rather than calling ActionContext.getSession().

The thread kicked off by this interceptor will be named in the form **actionNameBackgroundProcess**. For example, the *search* action would run as a thread named *searchBackgroundProcess*.

Parameters

- `threadPriority` (optional) - the priority to assign the thread. Default is `Thread.NORM_PRIORITY`.
- `delay` (optional) - an initial delay in millis to wait before the wait page is shown (returning `wait` as result code). Default is no initial delay.

- `delaySleepInterval` (optional) - only used with `delay`. Used for waking up at certain intervals to check if the background process is already done. Default is 100 millis.

Extending the Interceptor

If you wish to make special preparations before and/or after the invocation of the background thread, you can extend the `BackgroundProcess` class and implement the `beforeInvocation()` and `afterInvocation()` methods. This may be useful for obtaining and releasing resources that the background process will need to execute successfully. To use your background process extension, extend `ExecuteAndWaitInterceptor` and implement the `getNewBackgroundProcess()` method.

Examples

```

<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>
</pre>

<p/> <u>Example code2:</u>
This example will wait 2 second (2000 millis) before the wait page is shown to the user.
Therefore
if the long process didn't last long anyway the user isn't shown a wait page.

<pre>
<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>
</pre>

<p/> <u>Example code3:</u>
This example will wait 1 second (1000 millis) before the wait page is shown to the user.
And at every 50 millis this interceptor will check if the background process is done, if so
it will return before the 1 second has elapsed, and the user isn't shown a wait page.

<pre>
<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>
</pre>

```

```
</action>
</pre>
```

HibernateAndSpringEnabledExecuteAndWaitInterceptor

This page last changed on Oct 17, 2005 by [plightbo](#).

Find example code below for an extension of the *ExecuteAndWaitInterceptor*.

The goal of this code is to allow a background process to execute while having access to the same open Hibernate session object.

The SessionFactory dependency is injected into the *OpenSessionExecuteAndWaitInterceptor* by Spring. You may use other methods of dependency injection if you are more comfortable with them. By overriding the *getNewBackgroundProcess()* method, this interceptor uses our custom *OpenSessionBackgroundProcess* instead of the WebWork default.

Overriding the *beforeInvocation()* and *afterInvocation()* methods in the *OpenSessionBackgroundProcess* ensure that the session will stay open throughout the life of the background process, and any Spring transaction management will also be used.

As this code is heavily dependent on Spring and Hibernate, you shouldn't expect to see it packaged with a WebWork distribution. It does, however, serve as a useful example of extending the [Execute and Wait Interceptor](#)

```
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);
    }
}

```

Flash Interceptor

This page last changed on Dec 11, 2006 by [tm_jee](#).

Flash interceptor (FlashInterceptor) possibly with FlashResult allows current action to be available even after a redirect. It does this by saving the current action into http session and pushing it back into the stack next request, resulting in the nett effect of the action and its related information being available across redirect.

Parameters

- key - The Http Session key under which the action will be stored, default to FlashInterceptor#DEFAULT_KEY which is the string '__flashAction'.
- operation - The operation mode of this interceptor, either FlashInterceptor#STORE having a string value of 'Store' or FlashInterceptor#RETRIEVE having a string value of 'Retrieve' The default operation mode is FlashInterceptor#RETRIEVE

Extending the Interceptor

There's no intended extension points

Examples

```
<!-- Usage 1: (Using only Flash interceptor) -->
<action name="store" ...>
    <interceptor-ref name="flash">
        <param name="operation">Store</param>
    </interceptor-ref>
    <interceptor-ref name="defaultStack" />
    <result type="redirect">redirectToSomeWhere.jsp</result>
</action>
<action name="retrieve">
    <interceptor-ref name="flash">
        <param name="operation">Retrieve</param>
    </interceptor-ref>
    <interceptor-ref name="defaultStack" />
    <result>pageWhereWeNeedFlashActionStored.jsp</result>
</action>

<!-- Usage 2: (Using Flash Interceptor and Flash Result) -->
<action name="store">
    <result type="flash">redirectToSomeWhere.jsp</result>
</action>
<action name="retrieve">
    <interceptor-ref name="flash">
        <param name="operation">Retrieve</param>
    </interceptor-ref>
    <interceptor-ref name="defaultStack" />
    <result>pageWhereWeNeedFlashActionStored.jsp</result>
</action>
```

I18n Interceptor

This page last changed on Oct 18, 2005 by [digi9ten](#).

An interceptor that handles setting the locale specified in a session as the locale for the current action request. In addition, this interceptor will look for a specific HTTP request parameter and set the locale to whatever value is provided. This means that this interceptor can be used to allow for your application to dynamically change the locale for the user's session. This is very useful for applications that require multi-lingual support and want the user to be able to set his or her language preference at any point. The locale parameter is removed during the execution of this interceptor, ensuring that properties aren't set on an action (such as `request_locale`) that have no typical corresponding setter in your action.

For example, using the default parameter name, a request to `foo.action?request_locale=en_US`, then the locale for US English is saved in the user's session and will be used for all future requests.

Parameters

- `parameterName` (optional) - the name of the HTTP request parameter that dictates the locale to switch to and save in the session. By default this is **request_locale**
- `attributeName` (optional) - the name of the session key to store the selected locale. By default this is **WW_TRANS_I18N_LOCALE**

Extending the Interceptor

There are no known extensions points for this interceptor.

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 Oct 18, 2005 by [digi9ten](#).

This interceptor logs the start and end of the execution of an action (in English-only, not internationalized).

Parameters

There are no parameters for this interceptor.

Extending the Interceptor

There are no obvious extensions to the existing interceptor.

Examples

```
<!-- 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 Oct 18, 2005 by [digi9ten](#).

Watches for ModelDriven actions and adds the action's model on to the value stack.

Note: The ModelDrivenInterceptor must come before the both StaticParametersInterceptor and ParametersInterceptor if you want the parameters to be applied to the model.

Note: The ModelDrivenInterceptor will only push the model into the stack when the model is not null, else it will be ignored.

Parameters

- None

Extending the Interceptor

There are no known extension points to this interceptor.

Examples

```
<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 Feb 21, 2006 by [gjz22](#).

The Parameter Filter Interceptor blocks parameters from getting to the rest of the stack or your action. You can use multiple parameter filter interceptors for a given action, so, for example, you could use one in your default stack that filtered parameters you wanted blocked from every action and those you wanted blocked from an individual action you could add an additional interceptor for each action.

Parameters

- allowed - a comma delimited list of parameter prefixes that are allowed to pass to the action
- blocked - a comma delimited list of parameter prefixes that are not allowed to pass to the action
- defaultBlock - boolean (default to false) whether by default a given parameter is blocked. If true, then a parameter must have a prefix in the allowed list in order to be able to pass to the action

The way parameters are filtered for the least configuration is that if a string is in the allowed or blocked lists, then any parameter that is a member of the object represented by the parameter is allowed or blocked respectively.

For example, if the parameters are:

- blocked: person, person.address.createDate, personDao
- allowed: person.address
- defaultBlock: false

The parameters person.name, person.phoneNum etc would be blocked because 'person' is in the blocked list. However, person.address.street and person.address.city would be allowed because person.address is in the allowed list (the longer string determines permissions).

Parameter Remover Interceptor

This page last changed on Dec 04, 2006 by [tm_jee](#).

This is a simple WebWork interceptor that allows parameters (matching one of the paramNames attribute csv value) to be removed from the parameter map if they match a certain value (matching one of the paramValues attribute csv value), before they are set on the action. A typical usage would be to want a dropdown/select to map onto a boolean value on an action. The select had the options none, yes and no with values -1, true and false. The true and false would map across correctly. However the -1 would be set to false. This was not desired as one might needed the value on the action to stay null. This interceptor fixes this by preventing the parameter from ever reaching the action.

Parameters

- paramNames - A comma separated value (csv) indicating the parameter name whose param value should be considered that if they match any of the comma separated value (csv) from paramValues attribute, shall be removed from the parameter map such that they will not be applied to the action
- paramValues - A comma separated value (csv) indicating the parameter value that if matched shall have its parameter be removed from the parameter map such that they will not be applied to the action

Extending the Interceptor

No intended extension point

Examples

```
<action name="sample" class="org.martingilday.Sample">
    <interceptor-ref name="paramRemover">
        <param name="paramNames">aParam,anotherParam</param>
        <param name="paramValues">--, -1</param>
    </interceptor-ref>
    <interceptor-ref name="default-stack" />
    ...
</action>
```

Parameters Interceptor

This page last changed on Oct 20, 2005 by [plightbo](#).

This interceptor sets all parameters on the value stack. This interceptor gets all parameters from `ActionContext#getParameters()` and sets them on the value stack by calling `{@link OgnlValueStack#setValue(String, Object)}`, typically resulting in the values submitted in a form request being applied to an action in the value stack. Note that the parameter map must contain a String key and often containers a `String[]` for the value.

Because parameter names are effectively OGNL statements, it is important that security be taken in to account. This interceptor will not apply any values in the parameters map if the expression contains an assignment (`=`), multiple expressions `(.)`, or references any objects in the context `(#)`. This is all done in the `{@link #acceptableName(String)}` method. In addition to this method, if the action being invoked implements the `{@link ParameterNameAware}` interface, the action will be consulted to determine if the parameter should be set.

In addition to these restrictions, a flag (`XWorkMethodAccessor#DENY_METHOD_EXECUTION`) is set such that no methods are allowed to be invoked. That means that any expression such as `person.doSomething()` or `person.getName()` will be explicitly forbidden. This is needed to make sure that your application is not exposed to attacks by malicious users.

While this interceptor is being invoked, a flag (`InstantiatingNullHandler#CREATE_NULL_OBJECTS`) is turned on to ensure that any null reference is automatically created - if possible. See the type conversion documentation and the `InstantiatingNullHandler` javadocs for more information.

Finally, a third flag (`XWorkConverter#REPORT_CONVERSION_ERRORS`) is set that indicates any errors when converting the the values to their final data type (`String[] -> int`) an unrecoverable error occurred. With this flag set, the type conversion errors will be reported in the action context. See the type conversion documentation and the `XWorkConverter` javadocs for more information.

If you are looking for detailed logging information about your parameters, turn on DEBUG level logging for this interceptor. A detailed log of all the parameter keys and values will be reported.

For more information on ways to restrict the parameter names allowed, see the `ParameterNameAware` javadocs:

This interface is implemented by actions that want to declare acceptable parameters. Works in conjunction with `{@link ParametersInterceptor}`. For example, actions may want to create a whitelist of parameters they will accept or a blacklist of paramters they will reject to prevent clients from setting other unexpected (and possibly dangerous) parameters.

Parameters

- None

Extending the Interceptor

The best way to add behavior to this interceptor is to utilize the ParameterNameAware interface in your actions. However, if you wish to apply a global rule that isn't implemented in your action, then you could extend this interceptor and override the #acceptableName(String) method.

Examples

```
<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 Jun 11, 2006 by [tm_jee](#).

This interceptor calls prepare() on actions which implement Preparable. This interceptor is very useful for any situation where you need to ensure some logic runs before the actual execute method runs.

A typical use of this is to run some logic to load an object from the database so that when parameters are set they can be set on this object. For example, suppose you have a User object with two properties: *id* and *name*. Provided that the params interceptor is called twice (once before and once after this interceptor), you can load the User object using the *id* property, and then when the second params interceptor is called the parameter *user.name* will be set, as desired, on the actual object loaded from the database. See the example for more info.

In PrepareInterceptor

Applies only when action implements Preparable

1. if the action class have prepare{MethodName}(), it will be invoked
2. else if the action class have prepareDo{MethodName}{}(), it will be invoked
3. no matter if 1] or 2] is performed, if alwaysInvokePrepare property of the interceptor is "true" (which is by default "true"), prepare() will be invoked.

Parameters

- None

Extending the Interceptor

There are no known extension points to this interceptor.

Examples

```
<!-- 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 Oct 24, 2005 by [plightbo](#).

This is designed to solve a few simple issues related to wizard-like functionality in WebWork. One of those issues is that some applications have application-wide parameters commonly used, such `pageLen` (used for records per page). Rather than requiring that each action check if such parameters are supplied, this interceptor can look for specified parameters and pull them out of the session.

This works by setting listed properties at action start with values from session/application attributes keyed after the action's class, the action's name, or any supplied key. After action is executed all the listed properties are taken back and put in session or application context.

To make sure that each execution of the action is consistent it makes use of session-level locking. This way it guarantees that each action execution is atomic at the session level. It doesn't guarantee application level consistency however there has yet to be enough reasons to do so. Application level consistency would also be a big performance overkill.

Note that this interceptor takes a snapshot of action properties just before result is presented (using a {@link PreResultListener}), rather than after action is invoked. There is a reason for that: At this moment we know that action's state is "complete" as its values may depend on the rest of the stack and specifically - on the values of nested interceptors.

Parameters

- session - a list of action properties to be bound to session scope
- application - a list of action properties to be bound to application scope
- key - a session/application attribute key prefix, can contain following values:
 - CLASS - that creates a unique key prefix based on action namespace and action class, it's a default value
 - ACTION - creates a unique key prefix based on action namespace and action name
 - any other value is taken literally as key prefix
- type - with one of the following
 - start - means it's a start action of the wizard-like action sequence and all session scoped properties are reset to their defaults
 - end - means that session scoped properties are removed from session after action is run
 - any other value or no value means that it's in-the-middle action that is set with session properties before it's executed, and its properties are put back to session after execution
- sessionReset - boolean value causing all session values to be reset to action's default values or application scope values, note that it is similar to type="start" and in fact it does the same, but in our team it is sometimes semantically preferred. We use session scope in two patterns - sometimes there are wizard-like action sequences that have start and end, and sometimes we just want simply reset current session values.

Extending the Interceptor

There are no known extension points for this interceptor.

Examples

```
<!-- As the filter and orderBy parameters are common for all my browse-type actions,
you can move control to the scope interceptor. In the session parameter you can list
action properties that are going to be automatically managed over session. You can
do the same for application-scoped variables-->
<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 Oct 18, 2005 by [digi9ten](#).

An interceptor which sets action properties based on the interfaces an action implements. For example, if the action implements ParameterAware then the action context's parameter map will be set on it.

This interceptor is designed to set all properties an action needs if it's aware of servlet parameters, the servlet context, the session, etc. Interfaces that it supports are:

- ServletContextAware
- ServletRequestAware
- ServletResponseAware
- ParameterAware
- SessionAware
- ApplicationAware
- PrincipalAware

Parameters

- None

Extending the Interceptor

There are no known extension points for this interceptor.

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>
```

Session Validation Interceptor

This page last changed on Dec 04, 2006 by [tm_jee](#).

This interceptor invalidates http session based on the type of operation it is in. There's three type of operations:-

- NextRequest - This causes the interceptor to invalidate the session in the next comming request where this interceptor is present in the interceptor stack. This interceptor mark this in the http session using a key determined by the key attribute of this interceptor
- Now - This causes this interceptor to invalidate the session at the end of this interceptor's interception
- NoOperation - This causes this interceptor to basically do nothing. It is here such that users could have this interceptor in their default stack and still allows it to do nothing

Parameters

- type - indicate the operation of this interceptor, valid values are 'NextRequest', 'Now' and 'NoOperation' See description above for more information.
- key - this is the http session key used by the interceptor to mark the situation whereby the next comming request with this interceptor present in the interception stack, it will invalidate the http session.

Extending the Interceptor

No intended extension points.

Examples

```
<action name="logout" ... >
    <interceptor-ref name="sessionInvalidate">
        <param name="type">Now</param>
    </interceptor-ref>
    ...
</action>

or

<action name="sayByeByeNextRequestWillHaveSessionLost" ... >
    <interceptor-ref name="sessionInvalidate">
        <param name="type"><NextRequest/></param>
    </interceptor-ref>
    ...
</action>

<!-- This is the next request, "sessionInvalidate" will find the marker inserted
     by the action above and invalidate the session -->
<!-- The type="NoOperation" is just there so that the type is a valid one, and
     we don't get a warning log message -->
<action name="nextRequest" ... >
    <interceptor-ref name="sessionInvalidate">
```

```
<param name="type">NoOperation</param>
</interceptor-ref>
...
</action>
```

Static Parameters Interceptor

This page last changed on Oct 18, 2005 by [digi9ten](#).

This interceptor populates the action with the static parameters defined in the action configuration. If the action implements Parameterizable, a map of the static parameters will be also be passed directly to the action.

Parameters are typically defined with <param> elements within xwork.xml.

Parameters

- None

Extending the Interceptor

There are no extension points to this interceptor.

Examples

```
<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 Oct 18, 2005 by [digi9ten](#).

This interceptor logs the amount of time in milliseconds. In order for this interceptor to work properly, the logging framework must be set to at least the [INFO](#) level. This interceptor relies on the [Commons Logging API](#) to report its execution-time value.

Parameters

- `logLevel` (optional) - what log level should we use (`trace`, `debug`, `info`, `warn`, `error`, `fatal`)? - default is `info`
- `logCategory` (optional) - If provided we would use this category (eg. `com.mycompany.app`). Default is to use `com.opensymphony.xwork.interceptor.TimerInterceptor`.

The parameters above enables us to log all action execution times in our own logfile.

Extending the Interceptor

This interceptor can be extended to provide custom message format. Users should override the `invokeUnderTiming` method.

Examples

```
<!-- records only the action's execution time -->
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="completeStack"/>
    <interceptor-ref name="timer"/>
    <result name="success">good_result.ftl</result>
</action>

<!-- records action's execution time as well as other interceptors-->
<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 Oct 18, 2005 by [digi9ten](#).

Ensures that only one request per token is processed. This interceptor can make sure that back buttons and double clicks don't cause un-intended side affects. For example, you can use this to prevent careless users who might double click on a "checkout" button at an online store. This interceptor uses a fairly primitive technique for when an invalid token is found: it returns the result **invalid.token**, which can be mapped in your action configuration. A more complex implementation, TokenSessionStoreInterceptor, can provide much better logic for when invalid tokens are found.

Note: To set a token in your form, you should use the **token tag**. This tag is required and must be used in the forms that submit to actions protected by this interceptor. Any request that does not provide a token (using the token tag) will be processed as a request with an invalid token.

Internationalization Note: The following key could be used to internationalized the action errors generated by this token interceptor

- webwork.messages.invalid.token

NOTE: As this method extends off MethodFilterInterceptor, it is capable of deciding if it is applicable only to selective methods in the action class. See [MethodFilterInterceptor](#) for more info.

Parameters

- None

Extending the Interceptor

While not very common for users to extend, this interceptor is extended by the TokenSessionStoreInterceptor. The #handleInvalidToken and #handleValidToken methods are protected and available for more interesting logic, such as done with the token session interceptor.

Examples

```
<action name="someAction" class="com.examples.SomeAction">
    <interceptor-ref name="token"/>
    <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">
        <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 Oct 18, 2005 by [digi9ten](#).

This interceptor builds off of the TokenInterceptor, providing advanced logic for handling invalid tokens. Unlike the normal token interceptor, this interceptor will attempt to provide intelligent fail-over in the event of multiple requests using the same session. That is, it will block subsequent requests until the first request is complete, and then instead of returning the *invalid.token* code, it will attempt to display the same response that the original, valid action invocation would have displayed if no multiple requests were submitted in the first place.

NOTE: As this method extends off MethodFilterInterceptor, it is capable of deciding if it is applicable only to selective methods in the action class. See [MethodFilterInterceptor](#) for more info.

Parameters

- None

Extending the Interceptor

There are no known extension points for this interceptor.

Examples

```
<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 Oct 18, 2005 by [digi9ten](#).

This interceptor runs the action through the standard validation framework, which in turn checks the action against any validation rules (found in files such as *ActionClass-validation.xml*) and adds field-level and action-level error messages (provided that the action implements `com.opensymphony.xwork.ValidationAware`). This interceptor is often one of the last (or second to last) interceptors applied in a stack, as it assumes that all values have already been set on the action.

This interceptor does nothing if the name of the method being invoked is specified in the **excludeMethods** parameter. **excludeMethods** accepts a comma-delimited list of method names. For example, requests to **foo!input.action** and **foo!back.action** will be skipped by this interceptor if you set the **excludeMethods** parameter to "input, back".

Note that this has nothing to do with the `com.opensymphony.xwork.Validateable` interface and simply adds error messages to the action. The workflow of the action request does not change due to this interceptor. Rather, this interceptor is often used in conjunction with the **workflow** interceptor.

NOTE: As this method extends off `MethodFilterInterceptor`, it is capable of deciding if it is applicable only to selective methods in the action class. See `MethodFilterInterceptor` for more info.

Parameters

- None

Extending the Interceptor

There are no known extension points for this interceptor.

Examples

```
<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 Jun 11, 2006 by [tm_jee](#).

An interceptor that does some basic validation workflow before allowing the interceptor chain to continue.

This interceptor does nothing if the name of the method being invoked is specified in the **excludeMethods** parameter. **excludeMethods** accepts a comma-delimited list of method names. For example, requests to **foo!input.action** and **foo!back.action** will be skipped by this interceptor if you set the **excludeMethods** parameter to "input, back".

The order of execution in the workflow is:

1. If the action being executed implements Validateable, the action's Validateable#validate() validate method is called.
2. Next, if the action implements ValidationAware, the action's ValidationAware#hasErrors() hasErrors method is called. If this method returns true, this interceptor stops the chain from continuing and immediately returns Action#INPUT

Note: if the action doesn't implement either interface, this interceptor effectively does nothing. This interceptor is often used with the **validation** interceptor. However, it does not have to be, especially if you wish to write all your validation rules by hand in the validate() method rather than in XML files.

NOTE: As this method extends off MethodFilterInterceptor, it is capable of deciding if it is applicable only to selective methods in the action class. See [MethodFilterInterceptor](#) for more info.

Update: Added logic to execute a validate{MethodName} rather than a general validate method. This allows us to run some validation logic based on the method name we specify in the ActionProxy. For example, you can specify a validateInput() method, or even a validateDoInput() method that will be run before the invocation of the input method.

In DefaultWorkflowInterceptor

applies only when action implements com.opensymphony.xwork.Validateable

1. if the action class have validate{MethodName}(), it will be invoked
2. else if the action class have validateDo{MethodName}(), it will be invoked
3. no matter if 1] or 2] is performed, if alwaysInvokeValidate property of the interceptor is "true" (which is by default "true"), validate() will be invoked.

Parameters

- alwaysInvokeValidate - Default to true. If true validate() method will always be invoked, otherwise it will not.

Extending the Interceptor

There are no known extension points for this interceptor.

Examples

```
<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 Mar 21, 2006 by [phil](#).

Description

WebWork supports internationalization (in short, i18n) in two different places: the UI tags and the action/field error messages.

- [Tags](#) Typically the `i18n` and `text` tag
- [Validation](#)

Resource Bundle Search Order

Resource bundles are searched in the following order:

1. ActionClass.properties
2. BaseClass.properties (all the way to Object.properties)
3. Interface.properties (every interface and sub-interface)
4. ModelDriven's model (if implements ModelDriven), for the model object repeat from 1
5. package.properties (of the directory where class is located and every parent directory all the way to the root directory)
6. search up the i18n message key hierarchy itself
7. global resource properties (webwork.custom.i18n.resources) defined in webwork.properties

For more information, checkout the `LocalizedTextUtil` class.



Package Hierarchy

To clarify #5, while traversing the package hierarchy, WW will look for a file package.properties:
com/

```
acme/
  package.properties
  actions/
    package.properties
    FooAction.java
    FooAction.properties
```

If `FooAction.properties` does not exist, `com/acme/actions/package.properties` will be searched for, if not found `com/acme/package.properties`, if not found `com/package.properties`, etc.

Examples

Using `getText()`

To display i18n text, you can use a call to `getText()` in the [property](#) tag, or any other tag such as the UI tags (this is especially useful for labels of UI tags):

```
<ww:property value="getText('some.key')" />
```

Text Tag

You may also use the [text](#) tag:

```
<-- Third Example -->
<ww:text name="some.key" />

<-- Fourth Example -->
<a:text name="some.invalid.key" >
  The Default Message That Will Be Displayed
</a:text>
```

I18n Tag

Also, note that there is an [i18n](#) tag that will push a resource bundle on to the stack, allowing you to display text that would otherwise not be part of the resource bundle search hierarchy mentioned previously.

```
<ww:i18n name="some.package.bundle" >
  <ww:text name="some.key" />
</ww:i18n>
```



Internationalization in SiteMesh decorators is possible, but there are a few quirks about it. Check out the [SiteMesh](#) page to learn how to integrate WebWork and SiteMesh, including integration tips.

I18n Interceptor

See [I18n Interceptor](#) for more info. It basically pushes a locale into the ActionContext map upon every request. Webwork (components, ActionSupport etc.) honours this and hence every i18n related aspect will use this locale. Its a pretty elegant way of changing locale upon request as well.

Global resource (**webwork.custom.i18n.resources**) in **webwork.properties**

A global resource bundle could be specified through the 'webwork.custom.i18n.resources' property in webwork.properties. The locale can be switched by 'webwork.locale' in the webwork.properties as well.

Comparison with Struts

Struts users should be familiar with the application.properties resource bundle, where you can put all the messages in the application that are going to be translated. WebWork, though, splits the resource bundles per action or model class, and you may end up with duplicated messages in those resource bundles. A quick fix for that is to create a file called ActionSupport.properties in com/opensymphony/xwork and put it on your classpath. This will only work well if all your actions subclass ActionSupport.

Inversion of Control

This page last changed on Jan 03, 2007 by [phil](#).

What is Inversion of Control (IoC)?

Inversion of control is a way to handle dependencies of objects. For an overview of of Inversion of Control (also referred to now as Dependency Injection), please read [Martin Fowler's article on IoC](#).

Spring as recommended IoC Container

[Spring](#) is, among other things, an Inversion of Control framework. As of WebWork 2.2, it is the recommended IoC container. A detailed description how Spring Integration works and how it is configured is found [here](#).

Besides Spring, there are numerous other containers available for you to use, such as [Pico](#) or the (deprecated) integrated XWork IoC container.

WebWork/XWork integrated IoC Container



As of WebWork 2.2, the WebWork/XWork IoC container has been deprecated (though not removed) and the WebWork team recommends you use [Spring](#) for all your IoC needs

In WebWork IoC, objects that have their dependencies managed are called "components".

- [IoC Overview](#)
- [Xwork's Component Architecture](#)
- [How Webwork Uses Components](#)
- [Configuration of Components in Webwork and XWork](#)

Components

This page last changed on Oct 25, 2005 by [plightbo](#).



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 Aug 21, 2006 by [plightbo](#).



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:

```
<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 instances of 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 Oct 25, 2005 by [plightbo](#).



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

In many applications you have component objects that are required by a given class to use. In a nutshell, the IoC pattern allows a parent object (in the case of Webwork, XWork's ComponentManager instance) to give a resource Object to the action Object that needs it (usually an action, but it could be any object that implements the appropriate *enabler*) rather than said Object's needing to obtain the resource itself.

There are two ways of implementing IoC: Instantiation and using an enabler interface. With instantiation, a given action Object is instantiated with the resource Object as a constructor parameter. With enablers interfaces, the action will have an interface with a method, say "setComponent(ComponentObject r);", that will allow the resource to be passed to said action Object after it is instantiated. The ComponentObject is passed, because the Object implements the given interface. XWork uses *enablers* to pass components.

Why IoC?

So why is IoC useful? It means that you can develop components (generally services of some sort) in a top-down fashion, without the need to build a registry class that the client must then call to obtain the component instance.

Traditionally when implementing services you are probably used to following steps similar to these:

1. Write the component (eg an ExchangeRateService)
2. Write the client class (eg an XWork action)
3. Write a registry class that holds the component object (eg Registry)
4. Write code that gives the component object to the registry (eg Registry.registerService(new MyExchangeRateService()))
5. Use the registry to obtain the service from your client class (eg ExchangeRateService ers = Registry.getExchangeRateService())
6. Make calls to the component from the client class (eg String baseCurrencyCode = ers.getBaseCurrency())

Using IoC, the process is reduced to the following:

1. Write the component class (eg an ExchangeRateService)
2. Register the component class with XWork (eg componentManager.addEnabler(MyExchangeRateService, ExchangeRateAware))
3. Write the client class, making sure it implements the enabling interface (eg an XWork action that implements ExchangeRateAware)
4. Access the component instance directly from your client action (eg String baseCurrencyCode =

```
ers.getBaseCurrency())
```

More advantages of Inversion of Control are the following:

1. Testability - You can more easily test your objects by passing mock objects using the enabler method rather than needing to create full containers that allow your objects to get the components they need.
2. A component describes itself. When you instantiate a component, you can easily determine what dependencies it requires without looking at the source or using trial and error.
3. Dependencies can be discovered easily using reflection. This has many benefits ranging from diagram generation to runtime optimization (by determining in advance which components will be needed to fulfill a request and preparing them asynchronously, for example).
4. Avoids the super-uber-mega-factory pattern where all the components of the app are held together by a single class that is directly tied back to other domain specific classes, making it hard to 'just use that one class'.
5. Adheres to Law of Demeter. Some people think this is silly, but in practise I've found it works much better. Each class is coupled to only what it actually uses (and it should never use too much) and no more. This encourages smaller responsibility specific classes which leads to cleaner design.
6. Allows context to be isolated and explicitly passed around. ThreadLocals may be ok in a web-app, but they aren't well suited for high concurrency async applications (such as message driven applications).

Xwork's Component Architecture

This page last changed on Jan 07, 2006 by [plightbo](#).



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 its 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 explicitly 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 Mar 23, 2006 by [phil](#).

Using J2SE 5 ("Tiger") support with WebWork

Work in progress

This is a work in progress and not yet finished! More complex examples will follow. For now, have a look at the unit tests within the xwork-tiger project path.

To use J2SE 5 support with WebWork, you have to add xwork-tiger.jar to your classpath. The xwork-tiger.jar can be obtained via the ivy repository and is located in the lib/tiger directory.

Interceptor Annotations

To use these annotations, you have to specify the [AnnotationWorkflowInterceptor](#) to your interceptor stack.

Annotation	Description
After Annotation	Marks a action method that needs to be executed before the result.
Before Annotation	Marks a action method that needs to be executed before the main action method.
BeforeResult Annotation	Marks a action method that needs to be executed before the result.

Validation Annotations

If you want to use annotation based validation, you have to annotate the class or interface with [Validation Annotation](#).

These are the standard validator annotations that come with XWork-tiger:

Annotation	Description
ConversionErrorFieldValidator Annotation	Checks if there are any conversion errors for a field.
DateRangeFieldValidator Annotation	Checks that a date field has a value within a specified range.
DoubleRangeFieldValidator Annotation	Checks that a double field has a value within a specified range.
EmailValidator Annotation	Checks that a field is a valid e-mail address.

ExpressionValidator Annotation	Validates an expression.
FieldExpressionValidator Annotation	Uses an OGNL expression to perform its validator.
IntRangeFieldValidator Annotation	Checks that a numeric field has a value within a specified range.
RegexFieldValidator Annotation	Validates a regular expression for a field.
RequiredFieldValidator Annotation	Checks that a field is non-null.
RequiredStringValidator Annotation	Checks that a String field is not empty.
StringLengthFieldValidator Annotation	Checks that a String field is of the right length.
StringRegexValidator Annotation	
UrlValidator Annotation	Checks that a field is a valid URL.
Validation Annotation	Marker annotation for validation at Type level.
Validations Annotation	Used to group validation annotations.
VisitorFieldValidator Annotation	
CustomValidator Annotation	Use this annotation for your custom validator types.

Type Conversion Annotations

If the xwork-tigerjar is added to the classpath, you will directly have type conversion support for Maps and Collections using generics.

In short, instead of specifying the types found in collections and maps as documented in [Type Conversion](#), **the collection's generic type is used**. This means you most likely don't need any **ClassName-conversion.properties** files.

If you want to use annotation based type conversion, you have to annotate the class or interface with the [Conversion Annotation](#).

Annotation	Description
Conversion Annotation	Marker annotation for type conversions at Type level.
CreateIfNull Annotation	For Collection and Map types: Create the types within the Collection or Map, if null.
Element Annotation	For Generic types: Specify the element type for Collection types and Map values.
Key Annotation	For Generic types: Specify the key type for Map keys.
KeyProperty Annotation	For Generic types: Specify the key property name value.
TypeConversion Annotation	Used for class and application wide conversion rules.

Create **ClassName-conversion.properties** via "ant apt" target

This is an example for the apt ant target:

```
<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 Mar 07, 2007 by [maydeen](#).

After Annotation

Marks a action method that needs to be called after the main action method and the result was executed. Return value is ignored.

Usage

The After annotation can be applied at method level.

Parameters

no parameters

Examples

```
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;  
    }  
}
```

AnnotationWorkflowInterceptor

This page last changed on Mar 07, 2007 by [maydeen](#).

AnnotationWorkflowInterceptor Interceptor

Invokes any annotated methods on the action. Specifically, it supports the following annotations:

- @Before - will be invoked before the action method. If the returned value is not null, it is returned as the action result code
- @BeforeResult - will be invoked after the action method but before the result execution
- @After - will be invoked after the action method and result execution

There can be multiple methods marked with the same annotations, but the order of their execution is not guaranteed. However, the annotated methods on the superclass chain are guaranteed to be invoked before the annotated method in the current class in the case of a Before annotations and after, if the annotations is After.

Examples

```
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";  
    }  
}
```

Configure a stack in xwork.xml that replaces the PrepareInterceptor with the AnnotationWorkflowInterceptor:

```
<interceptor-stack name="annotatedStack">
<interceptor-ref name="static-params"/>
<interceptor-ref name="params"/>
<interceptor-ref name="conversionError"/>
<interceptor-ref name="annotationInterceptor"/>
</interceptor-stack>
```

Given an Action, AnnotatedAction, add a reference to the AnnotationWorkflowInterceptor interceptor.

```
<action name="AnnotatedAction" class="com.examples.AnnotatedAction">
  <interceptor-ref name="annotationInterceptor"/>
  <result name="success" type="freemarker">good_result.ftl</result>
</action>
```

With the interceptor applied and the action executed on AnnotatedAction the log instance variable will contain baseBefore-before-execute-beforeResult-after.

Before Annotation

This page last changed on Mar 07, 2007 by [maydeen](#).

Before Annotation

Marks a action method that needs to be executed before the main action method.

Usage

The Before annotation can be applied at method level.

Parameters

no parameters

Examples

```
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;  
    }  
}
```

BeforeResult Annotation

This page last changed on Mar 07, 2007 by [maydeen](#).

BeforeResult Annotation

Marks a action method that needs to be executed before the result. Return value is ignored.

Usage

The BeforeResult annotation can be applied at method level.

Parameters

no parameters

Examples

```
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;  
    }  
}
```

Conversion Annotation

This page last changed on Mar 07, 2007 by [maydeen](#).

Conversion Annotation

A marker annotation for type conversions at Type level.

Usage

The Conversion annotation must be applied at Type level.

Parameters

Parameter	Required	Default	Description
conversion	no		used for Type Conversions applied at Type level.

Examples

```
@Conversion()  
public class ConversionAction implements Action {  
}
```

ConversionErrorFieldValidator Annotation

This page last changed on Mar 07, 2007 by [maydeen](#).

ConversionErrorFieldValidator Annotation

This validator checks if there are any conversion errors for a field and applies them if they exist. See [Type Conversion Error Handling](#) for details.

Usage

The ConversionErrorFieldValidator annotation must be applied at method level.

Parameters

Parameter	Required	Default	Notes
message	yes		field error message
key	no		i18n key from language specific properties file.
fieldName	no		
shortCircuit	no	false	If this validator should be used as shortCircuit.
type	yes	ValidatorType.FIELD	Enum value from ValidatorType. Either FIELD or SIMPLE can be used here.

Examples

```
@ConversionErrorFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true)
```

CreateIfNull Annotation

This page last changed on Mar 02, 2006 by [rainerh](#).

CreateIfNull Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

CustomValidator Annotation

This page last changed on Mar 23, 2006 by [rainerh](#).

CustomValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Adding Parameters

Use the [ValidationParameter annotation](#) to add custom parameter values.

ValidationParameter annotation

This page last changed on Mar 23, 2006 by [rainerh](#).

ValidationParameter Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

DateRangeFieldValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

DateRangeFieldValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

DoubleRangeFieldValidator Annotation

This page last changed on Mar 02, 2006 by [rainerh](#).

DoubleRangeFieldValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Element Annotation

This page last changed on Mar 02, 2006 by [rainerh](#).

Element Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

EmailValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

EmailValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

ExpressionValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

ExpressionValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

FieldExpressionValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

FieldExpressionValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

IntRangeFieldValidator Annotation

This page last changed on Dec 19, 2006 by [mrdon](#).

IntRangeFieldValidator Annotation

This validator checks that a numeric field has a value within a specified range. If neither min nor max is set, nothing will be done.

Usage

The annotation must be applied at method level.

Parameters

Parameter	Required	Default	Notes
message	yes		field error message
key	no		i18n key from language specific properties file.
fieldName	no		
shortCircuit	no	false	If this validator should be used as shortCircuit.
type	yes	ValidatorType.FIELD	Enum value from ValidatorType. Either FIELD or SIMPLE can be used here.
min	no		Integer property. The minimum the number must be.
max	no		Integer property. The maximum number can be.

If neither *min* nor *max* is set, nothing will be done.

The values for min and max must be inserted as String values so that "0" can be handled as a possible value.

Examples

```
@IntRangeFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true, min  
= "0", max = "42")
```

Key Annotation

This page last changed on Mar 07, 2007 by [maydeen](#).

Key Annotation

Sets the Key for type conversion.

Usage

The Key annotation must be applied at field or method level.

Parameters

Parameter	Required	Default	Description
value	no	java.lang.Object.class	The key property value.

Examples

```
// 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;
```

KeyProperty Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

KeyProperty Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

RegexFieldValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

RegexFieldValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

RequiredFieldValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

RequiredFieldValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

RequiredStringValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

RequiredStringValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

StringLengthFieldValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

StringLengthFieldValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

StringRegexValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

StringRegexValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

TypeConversion Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

TypeConversion Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

UrlValidator Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

UrlValidator Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Validation Annotation

This page last changed on Mar 07, 2007 by [maydeen](#).

Validation Annotation

If you want to use annotation based validation, you have to annotate the class or interface with Validation Annotation.

Usage

The Validation annotation must be applied at Type level.

Parameters

Parameter	Required	Default	Notes
message	yes		field error message
key	no		i18n key from language specific properties file.
fieldName	no		
shortCircuit	no	false	If this validator should be used as shortCircuit.
type	yes	ValidatorType.FIELD	Enum value from ValidatorType. Either FIELD or SIMPLE can be used here.

Examples

An Annotated Interface

- Mark the interface with @Validation()
- Apply standard or custom annotations at method level

```
@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();
}

```

An Annotated Class

```

@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;
    }
}

```

Validations Annotation

This page last changed on Dec 12, 2005 by [rainerh](#).

Validations Annotation

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Usage

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Parameters

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

Examples

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

VisitorFieldValidator Annotation

This page last changed on Apr 06, 2007 by [maydeen](#).

VisitorFieldValidator Annotation

The validator allows you to forward validator to object properties of your action using the objects own validator files. This allows you to use the ModelDriven development pattern and manage your validations for your models in one place, where they belong, next to your model classes.

The VisitorFieldValidator can handle either simple Object properties, Collections of Objects, or Arrays. The error message for the VisitorFieldValidator will be appended in front of validator messages added by the validations for the Object message.

Usage

The annotation must be applied at method level.

Parameters

Parameter	Required	Default	Notes
message	yes		field error message
key	no		i18n key from language specific properties file.
fieldName	no		
shortCircuit	no	false	If this validator should be used as shortCircuit.
type	yes	ValidatorType.FIELD	Enum value from ValidatorType. Either FIELD or SIMPLE can be used here.
context	no	action alias	Determines the context to use for validating the Object property. If not defined, the context of the Action validation is propagated to the Object property validation. In the case of Action validation, this context is the Action alias.
appendPrefix	no	true	Determines whether the field name of this field

		<p>validator should be prepended to the field name of the visited field to determine the full field name when an error occurs. For example, suppose that the bean being validated has a "name" property. If <i>appendPrefix</i> is true, then the field error will be stored under the field "bean.name". If <i>appendPrefix</i> is false, then the field error will be stored under the field "name".</p> <p> If you are using the VisitorFieldValidator to validate the model from a ModelDriven Action, you should set <i>appendPrefix</i> to false unless you are using "model.name" to reference the properties on your model.</p>
--	--	--

Examples

```
@VisitorFieldValidator(message = "Default message", key = "i18n.key", shortCircuit = true,
context = "action alias", appendPrefix = true)
```

JasperReports

This page last changed on Mar 21, 2006 by [phil](#).

Introduction

JasperReports (<http://jasperreports.sourceforge.net>) is one of the leading open-source java reporting libraries. It compiles .jrxml (XML source) to .jasper (=compiled version) files, which in turn can be transformed to several output types (PDF, CSV, XLS and HTML).

In the following example, we will use Webwork to dynamically create a PDF with a list of persons. Our WW action will be used to create a List with objects, and our JasperReport Result will use this list to fill our template, and return the PDF.

I assume you already have a WW webapp working.



Note: although this is a very simple example, I suggest you read the fine documentation of both WW and JR.



Used versions

Webwork 2.2 beta 3 (but should apply to previous versions)
JasperReports 1.1.0
JDK 1.4.2

Right, let's begin.

Our Person class

We start by defining a simple POJO class: 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;
}

}

```

Nothing special. Just your basic properties, constructor, getters and setters.

JasperReports libraries

Before we can continue, we need to add the JR libraries to our classpath. You can download the JR project here: <http://www.sourceforge.net/projects/jasperreports>

Save the jasperreports-X-project.zip to your harddisk, and extract the files.

We need the following files:

- dist/jasperreports-X.jar
- lib/commons-*jar (all the commons - except maybe for commons-logging)
- lib/itext-X.jar
- lib/jdt-compiler.jar

Copy these jars over to your WW_WEBAPP/WEB-INF/lib directory, and add them to your classpath.

Show me the 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", "Carrolla");
        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;
    }
}
```

Once again - I guess everything is pretty clear. Our JasperAction will create a list of several People. The JasperCompileManager will then compile the jrxml template to a .jasper file.

 Again, don't use this in production code. You should of course either provide compiled templates, or do some sort of checking to avoid compiling the template on every request. But for our demonstration, or development, this suits our needs just fine.

Our Jasper template

JR uses a special XML page to define templates which will be compiled to .jasper files. These templates will be used to design the resulting report. It's pretty straightforward.

This is a handwritten version - for more complex versions I seriously suggest taking a look at the various GUI designers.

```
<?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>
        </band>
    </detail>

```

```

</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>

```

Save this file in WW_WEBAPP/jasper/ as 'our_jasper_template.jrxml'.

Most important: we declared the fields name and lastName (not surprisingly, two properties from our Person.class). This means we will now be able to use these fields in our Jasper template.

We define two columnheaders (NAME and LASTNAME), and then add our fields to the detail band (for a better explanation, look at the JR tutorial). This 'detail' band will iterate over our List of People. This is the default behaviour of JR - so if you want to display more information from the Person, add them to this band.

In the detail band we use the

`$F{name}`

expression. This means JR will ask WW how to retrieve the field value. WW will happily look up this value in the stack (find the person, and invoke the getName() getter), and return it. Similar for the

`$F{lastName}`

The rest is mostly markup to define the layout.



- Use a logger (commons-logging, log4j, ..) to watch com.opensymphony.webwork.views.jasperreports in debug mode, if you have any troubles

Register the Action

Alright, time to add the action to our xwork.xml file:

```
<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>
```

Let's explore this a bit further. I assume you are familiar with the xwork notation & schema, if not, check the documentation.

```
<action name="myJasperTest" class="com.mevipro.test.action.JasperAction">
```

We simply register our JasperAction with the name 'myJasperTest' - this means that we can execute this Action by sending a request to myJasperTest.action in our browser.

```
<result name="success" type="jasper">
```

When our JasperAction executes correctly, we will use the Result type registered with the name 'jasper'. This is already done when you include the webwork-default (

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

). This result type will be configured by our params, which we'll specify below:

```
<param name="location">/jasper/our_compiled_template.jasper</param>
```

This parameter defines the location of our compiled jasper file, which will be filled by WW with our dataSource:

```
<param name="dataSource">myList</param>
```

The name of the dataSource - this is the name of the getter you want to call (this will result in a getMyList() call to your JasperAction). It will be used to fill the template with data.

```
<param name="format">PDF</param>
```

This specifies the format to which the jasper file will be transformed. Possible values are: PDF, CSV, XLS and HTML.

Conclusion

You should now be able to execute http://localhost:8080/YOUR_WEBAPP/myJasperTest.action - and you should see a nice list of names.

WW provides probably the most elegant way to deal with JasperReport files; specify the location of the .jasper file, specify what dataSource you want to use, and there you go.

JSP

This page last changed on Sep 18, 2006 by [phil](#).

JSP support in WebWork is very easy: by default [webwork-default.xml](#) configures the [Dispatcher Result](#) as the default result (see [Result Types](#)). This means any JSP 1.2+ container can work with WebWork immediately.

Getting Started

Because JSP support occurs through the [Dispatcher Result](#), which is the default result type, you don't need to specify the type attribute when configuration [xwork.xml](#):

```
<action name="test" class="com.acme.TestAction">
    <result name="success">test-success.jsp</result>
</action>
```

Then in **test-success.jsp**:

```
<%@ taglib prefix="ww" uri="/webwork" %>

<html>
<head>
    <title>Hello</title>
</head>
<body>

Hello, <ww:property value="name"/>

</body>
</html>
```

Where **name** is a property on your action. That's it!

Tag Support

See the [JSP Tags](#) documentation for information on how to use the generic [Tags](#) provided by WebWork.

JUnit

This page last changed on Jun 09, 2006 by [phil](#).

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.AreEqual;
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.AreEqual(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();
    }
}

```

Test validation for an action

This is an example of how to simply test validation when using xml validation files. The following code would test the contents of the xml file "ValidateableAction-contextName-validation.xml".

```

public void testActionValidation() throws ValidationException {
    DefaultActionValidatorManager validator = new DefaultActionValidatorManager();
    ValidateableAction action = new ValidateableAction();
    action.setParam(new Param());
    ... // setup the parameters you want to test

    validator.validate(action, "contextName");
    assertTrue(action.hasErrors()); // If the validation should fail
    assertFalse(action.hasErrors()); // If the validation should pass
    // action.getActionErrors() will give you the errors that occurred
}

```

Note: these are not the ONLY ways so make your own judgement.

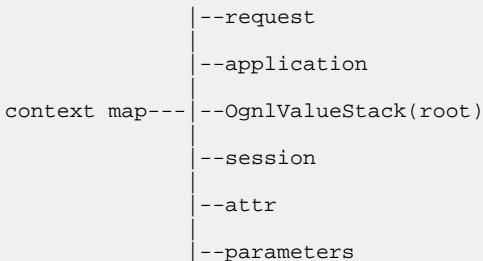
OGNL

This page last changed on Sep 16, 2006 by [phil](#).

Overview

OGNL is the Object Graph Navigation Language - see <http://wwwognl.org> for the full documentation of OGNL. In this document we will only show a few examples of OGNL features that co-exist with Webwork. To review basic concepts, refer to [OGNL Basics](#).

Webwork uses a standard naming context to evaluate OGNL expressions. The top level object dealing with OGNL is a map (usually referred as a context map). OGNL has a concept of a root object (in webwork terms, this is the OGNLValueStack). Along with the root, other objects are placed in the context map (referred as in the context) including your session/application/request/attr maps. These objects have nothing to do with the root, they just exist along side it in the context map. So, to access these objects, the # is used telling ognl not to look in the root object, but within the rest of the context



Note that there are other objects in the context map, I'm just referring to a few for this example. Now, your actions instances are placed in the OGNLValueStack so you can refer to your bean properties without the #.

```
<ww: property value="myBean.myProperty" />
```

For sessions,request, and the rest that lie in the context map:

```
ActionContext.getContext().getSession().put("mySessionPropKey", mySessionObject);  
  
<ww:property value="#session.mySessionPropKey"/> or  
<ww:property value="#session['mySessionPropKey']"/> or  
<ww:property value="#attr.mySessionPropKey"/>
```



A note about #attr

If a page context is present, #attr will search the page context. else it #attr will search request, session then finally application scope for the attribute.

Collections (Maps, Lists, Sets)

Dealing with collections(maps, lists, and sets) in webwork comes often, so here are a few examples using the select tag:

Syntax for list: {e1,e2,e3}. This creates a List containing the String "name1", "name2" and "name3". It also selects "name2" as the default value. Notice the use of the [Alt Syntax](#) to provide the literal "name2".

```
<ww:select label="label" name="name" list="{'name1', 'name2', 'name3'}" value="%{'name2'}" />
```

Syntax for map: #{key1:value1,key2:value2}. This creates a map that maps the string "foo" to the string "foovalue" and "bar" to the 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`

```
<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>
```

To select a subset of a collection (called projection), you can use a wildcard within the collection.

- ? - All elements matching the selection logic
- ^ - Only the first element matching the selection logic
- \$ - Only the last element matching the selection logic

To obtain a subset of just male relatives from the object person:

```
person.relatives.{? #this.gender == 'male'}
```

Lambda Expressions

OGNL supports basic lambda expression syntax enabling you to write simple functions. For example:

For all you math majors who didn't think you would ever see this one again.

Fibonacci: 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

Useful Information

The lambda expression is everything inside the brackets. The #this variable holds the argument to the expression, which is initially starts at 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 Dec 15, 2005 by [digi9ten](#).

XWork-specific language features

The biggest addition that XWork provides on top of OGNL is the support for the ValueStack. While OGNL operates under the assumption there is only one "root", XWork's ValueStack concept requires there be many "roots".

For example, suppose we are using standard OGNL (not using XWork) and there are two objects in the OgnlContext map: "foo" -> foo and "bar" -> bar and that the foo object is also configured to be the single **root** object. The following code illustrates how OGNL deals with these three situations:

```
#foo.blah // returns foo.getBlah()  
#bar.blah // returns bar.getBlah()  
blah      // returns foo.getBlah() because foo is the root
```

What this means is that OGNL allows many objects in the context, but unless the object you are trying to access is the root, it must be prepended with a namespaces such as @bar. Now let's talk about how XWork is a little different...

Useful Information

In XWork, the entire ValueStack is the root object in the context. Rather than having your expressions get the object you want from the stack and then get properties from that (ie: peek().blah), XWork has a special OGNL PropertyAccessor that will automatically look at all entries in the stack (from the top down) until it finds an object with the property you are looking for.

For example, suppose the stack contains two objects: Animal and Person. Both objects have a "name" property, Animal has a "species" property, and Person has a "salary" property. Animal is on the top of the stack, and Person is below it. The follow code fragments help you get an idea of what is going on here:

```
species    // call to animal.getSpecies()  
salary     // call to person.getSalary()  
name       // call to animal.getName() because animal is on the top
```

In the last example, there was a tie and so the animal's name was returned. Usually this is the desired effect, but sometimes you want the property of a lower-level object. To do this, XWork has added support for indexes on the ValueStack. All you have to do is:

```
[0].name   // call to animal.getName()  
[1].name   // call to person.getName()
```

With expression like [0] ... [3] etc. WebWork will cut the stack and still return back a CompoundRoot object. To get the top of that particular stack cut, use 0.top

ognl expression	description
[0].top	would get the top of the stack cut starting from element 0 in the stack (similar to top in this case)
[1].top	would get the top of the stack cut starting from element 1 in the stack

Accessing static properties

OGNL supports accessing static properties as well as static methods. As the OGNL docs point out, you can explicitly call statics by doing the following:

```
@some.package.ClassName@FOO_PROPERTY  
@some.package.ClassName@someMethod()
```

However, XWork allows you to avoid having to specify the full package name and call static properties and methods of your action classes using the "vs" prefix:

```
@vs@FOO_PROPERTY  
@vs@someMethod()  
  
@vs1@FOO_PROPERTY  
@vs1@someMethod()  
  
@vs2@BAR_PROPERTY  
@vs2@someOtherMethod()
```

"vs" stands for "value stack". The important thing to note here is that if the class name you specify is just "vs", the class for the object on the top of the stack is used. If you specify a number after the "vs" string, an object's class deeper in the stack is used instead.

Differences from the WebWork 1.x EL

Besides the examples and descriptions given above, there are a few major changes in the EL since WebWork 1.x. The biggest one is that 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. Because XWork is not built only for the web, there is no concept of "request attributes", and thus "#foo" is merely a request to another object in the OgnlContext other than the root.

Old Expression	New Expression
foo/blah	foo.blah
foo/someMethod()	foo.someMethod()
../bar/blah	[1].bar.blah
@baz	not directly supported, but #baz is similar

.	top or [0]
---	------------

WebWork-specific named objects

name	value
#parameters['foo'] or #parameters.foo	request parameter ['foo'] (request.getParameter())
#request['foo'] or #request.foo	request attribute ['foo'] (request.getAttribute())
#session['foo'] or #session.foo	session attribute 'foo'
#application['foo'] or #application.foo	ServletContext attributes 'foo'
#attr['foo'] or #attr.foo	Access to PageContext if available, otherwise searches request/session/application respectively

PreResultListener

This page last changed on Dec 06, 2006 by [phil](#).

PreResultListener is a little hook that results in a callback being made to it before the actual result is to be executed but after the action invocation took place.

How does it works

- start request
 - begin of interception 1
 - begin of interception 2
 - action
 - pre-result-listener kicks in
 - execution of result
 - end of interception 2
 - end of interception 1
- end request

How to add PreResultListener in my code?

Through Interceptor

```
public class MyInterceptor implements Interceptor {  
    ...  
    public String intercept(ActionInvocation invocation) throws Exception {  
        // we could hook up PreResultListener here  
        invocation.addPreResultListener(new PreResultListener() {  
            public void beforeResult(ActionInvocation invocation, String result) {  
                ...  
            }  
        });  
    }  
    ...  
}
```

Through Action

```
public class MyAction extends ActionSupport {  
    public String execute() throws Exception {  
        ActionContext.getContext().getActionInvocation().addPreResultListener() {  
            new PreResultListener() {  
                public void beforeResult(ActionInvocation invocation, String result) {  
                    ...  
                }  
            };  
        };  
    }  
}
```

Result Types

This page last changed on Feb 14, 2007 by [tm_jee](#).

See [Result Configuration](#) for basic information about how results are configuration.

Overview

Result Types are classes that determine what happens after an Action executes and a Result is returned. Developers are free to create their own Result Types according to the needs of their application or environment. In WebWork 2 for example, Servlet and Velocity Result Types have been created to handle rendering views in web applications.

Note: All built in webwork result types implement the `com.opensymphony.xwork.Result` interface, which represents a generic interface for all action execution results, whether that be displaying a webpage, generating an email, sending a JMS message, etc.

Result types define classes and map them to names to be referred in the action configuration results. This serves as a shorthand name-value pair for these classes.

```
...
<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>
```

Writing Custom Results

To write a custom Result, one would need to either

- implements com.opensymphony.xwork.Result interface
- extends com.opensymphony.webwork.dispatcher.WebWorkResultSupport

Implementing Result interface



When writing Web based application that uses WebWork and XWork, it is advised to extends custom Result from WebWorkResultSupport in favour of implementing Result interface as there are quite a bit of functionality that WebWork could recognized from the former approach.

Result interface is pretty straight forward with just the following method to be implemented

```
public void execute(ActionInvocation invocation) throws Exception;
```

The above execute(...) method is called when a WebWork action is being executed, a custom implementation could do thing like eg. sending an email, sending a JMS message, rendering some Swing component or just dispatching to a JSP page.

Various information could be obtained from the ActionInvocation object passed in as the argument.

Information	Method
WebWork's Action Object	actionInvocation.getAction()
Has Action Invocation being executed before	actionInvocation.isExecuted()
WebWork's Action context	actionInvocation.getInvocationContext()
WebWork's Action Proxy	actionInvocation.getProxy()
WebWork's Ognl Value Stack	actionInvocation.getStack()

Extending WebWorkResultSupport

WebWorkResultSupport is a support Result class that implements Result interface and add on various convenience methods that might prove useful to its subclass. The information regarding Result interface applies to WebWorkResultSupport as well. WebWorkSupportResult adds the following functionality:-

location property and its relevant accessors

With this property present, WebWork will be able to determine the location of a result. eg. instead of writting xwork's xml descriptor like this

```
<action ...>
  <result ...>
    <param name="location">xxxx</param>
  </result>
</action>
```

one could just do

```
<action ...>
<result ...>xxxx</result>
</action>
```

as without the present of <param> tag, the content of the <result> tag will be taken as the location property.

parse property and its relevant accessors

WebWorkResultSupport added a parse property which by default is true, which could be disabled by eg.

```
<action ...>
<result ...>
  <param name="parse">false</param>
  ...
</result>
</action>
```

This enable result's parameter to be parsed with respect to Ognl's value stack. and returns its String representation

```
<action ...>
<result ...>${protocol}${host}:${port}${contextPath}/xxxx/yyy.jsp</result>
</action>
```

With an action having the following properties being in the stack :-

Property	Value
protocol	http://
host	localhost
port	8080
contextPath	/context

The "location" property of the above result will be effectively, <http://localhost:8080/context/xxxx/yyy.jsp>



The parsing is delegated to WebWork's TextParseUtil class

encode property and its relevant accessors

WebWorkResultSupport added an encode property which by default is false which could be activated by :-

```
<action ...>
<result ...>
  <param name="encode">true</param>
  ...
</result>
</action>
```

This enable result's parameter to be encoded (using URLEncoder.encode("....", "UTF-8")) if it is set to true.

Result Types

Webwork provides several implementations of the `com.opensymphony.xwork.Result` interface to make web-based interactions with your actions simple. These result types include:

- [Chain Result](#) - used for [Action Chaining](#)
- [Dispatcher Result](#) - used for [JSP](#) integration
- [FreeMarker Result](#) - used for [FreeMarker](#) integration
- [HttpHeader Result](#) - used to control special HTTP behaviors
- [JasperReports Result](#) - used for [JasperReports](#) integration
- [Redirect Result](#) - used to redirect to another URL
- [Redirect Action Result](#) - used to redirect to another action
- [Stream Result](#) - used to stream an InputStream back to the browser (usually for file downloads)
- [Velocity Result](#) - used for [Velocity](#) integration
- [XSL Result](#) - used for XML/XSLT integration
- [PlainText Result](#) - used to display the raw content of a particular page (eg. jsp, html etc)
- [Flash Result](#) - used to store current action into http session such that it could be retrieved and push into the stack in subsequent request through FlashInterceptor.

Results are specified in a xwork xml config file(xwork.xml) nested inside `<action>`. If the `location` param is the only param being specified in the result tag, you can simplify it as follows:

```
<action name="bar" class="myPackage.barAction">
    <result name="success" type="dispatcher">
        <param name="location">foo.jsp</param>
    </result>
</action>
```

or simplified

```
<action name="bar" class="myPackage.barAction">
    <result name="success" type="dispatcher">foo.jsp</result>
</action>
```

if you are extending `webwork-default.xml`, then the default result type is "dispatcher". Also, if you don't specify the name of a result, it is assumed to be "success". This means you can simply the result down to

```
<action name="bar" class="myPackage.barAction">
    <result>foo.jsp</result>
</action>
```

NOTE: The `Parse` attribute enables the `location` element to be parsed for expressions. An example of how this could be useful:

```
<result name="success" type="redirect">/displayCart.action?userId=${userId}</result>
```

NOTE: You can also specify global-results to use with multiple actions. This can save time from having to add the same result to many different actions. For more information on result tags and global-results, see [Result Configuration](#) section.

Chain Result

This page last changed on Dec 11, 2005 by [plightbo](#).

This result invokes an entire other action, complete with it's own interceptor stack and result.

Parameters

- **actionName (default)** - the name of the action that will be chained to
- **namespace** - used to determine which namespace the Action is in that we're chaining. If namespace is null, this defaults to the current namespace
- **method** - used to specify another method on target action to be invoked. If null, this defaults to execute method

Examples

```
<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 Dec 20, 2005 by [mrdon](#).

Includes or forwards to a view (usually a jsp). Behind the scenes WebWork will use a RequestDispatcher, where the target servlet/JSP receives the same request/response objects as the original servlet/JSP. Therefore, you can pass data between them using `request.setAttribute()` - the WebWork action is available. There are three possible ways the result can be executed:

- If we are in the scope of a JSP (a PageContext is available), PageContext's {@link PageContext#included(String) include} method is called.
- If there is no PageContext and we're not in any sort of include (there is no "javax.servlet.include.servlet_path" in the request attributes), then a call to {@link RequestDispatcher#forward(javax.servlet.ServletRequest, javax.servlet.ServletResponse) forward} is made.
- Otherwise, {@link RequestDispatcher#include(javax.servlet.ServletRequest, javax.servlet.ServletResponse) include} is called.

Parameters

- **location (default)** - the location to go to after execution (ex. jsp).
- **parse** - true by default. If set to false, the location param will not be parsed for Ognl expressions.

Examples

```
<result name="success" type="dispatcher">
  <param name="location">foo.jsp</param>
</result>
```

Flash Result

This page last changed on Dec 11, 2006 by [tm_jee](#).

A flash result, that save the current action into the http session before invoking `super.doExecute(...)`, which actually just do a redirect to a specific location just as a normal `ServletRedirectResult` would.

Parameters

key - The key under which current action is stored in Http Session. Default to `FlashInterceptor#DEFAULT_KEY` which is the string '`__flashAction`'

Examples

```
<action name="store">
    <result type="flash"></redirectToSomeWhere.jsp</result>
</action>
<action name="retrieve">
    <interceptor-ref name="flash">
        <param name="operation">Retrieve</param>
    </interceptor-ref>
    <interceptor-ref name="defaultStack" />
    <result>pageWhereWeNeedFlashActionStored.jsp</result>
</action>
```

FreeMarker Result

This page last changed on Dec 20, 2005 by [mrdon](#).

Renders a view using the Freemarker template engine. Alternatively, the com.opensymphony.webwork.dispatcher.ServletDispatcherResult dispatcher result type can be used in conjunction Webwork's {@link FreemarkerServlet}.

The FreemarkerManager class configures the template loaders so that the template location can be either

- relative to the web root folder. eg /WEB-INF/views/home.ftl
- a classpath resource. eg com/company/web/views/home.ftl

Also see [WebWork Freemarker Support](#).

Parameters

- **location (default)** - the location of the template to process.
- **parse** - true by default. If set to false, the location param will not be parsed for Ognl expressions.
- **contentType** - defaults to "text/html" unless specified.

Examples

```
<result name="success" type="freemarker">foo.ftl</result>
```

WebWork Freemarker Support

This page last changed on Nov 30, 2004 by [plightbo](#).

Freemarker Support in WebWork

Freemarker views can be rendered either via the webwork result type `freemarker`, or by using the `dispatcher` result type in conjunction Webwork's `FreemarkerServlet`.

This document will focus on using the `freemarker` result since it is the recommended approach. An section follows to show how to use the `FreemarkerServlet`.

Configure your action to use the freemarker result type

The `freemarker` result type is defined in `webwork-default.xml`, so normally you just include it, and define your results to use `type="freemarker"`.

```
<include file="webwork-default.xml"/>
...
<action name="test" class="package.Test">
    <result name="success" type="freemarker">/WEB-INF/views/testView.ftl</result>
</action>
...
```

Property Resolution

Your action properties are automatically resolved - just like in a velocity view.

for example `${name}` will result in `stack.findValue("name")`, which *generally* results in `action.getName()` being executed.

A search process is used to resolve the variable, searching the following scopes in order, until a value is found :

- freemarker variables
- value stack
- request attributes
- session attributes
- servlet context attributes

Objects in the Context

The following variables exist in the freemarker views

- `req` - the current `HttpServletRequest`

- `res` - the current `HttpServletResponse`
- `stack` - the current `OgnlValueStack`
- `ognl` - the `OgnlTool` instance
 - This class contains useful methods to execute OGNL expressions against arbitrary objects, and a method to generate a select list using the `<ww:select>` pattern. (i.e. taking the name of the list property, a listKey and listValue)
- `webwork` - an instance of `FreemarkerWebWorkUtil`
- `action` - the current WebWork action
- `exception` - *optional* the `Exception` instance, if the view is a JSP exception or Servlet exception view

FreeMarker configuration with recent (post 2.1) releases

To configure the freemarker engine that webwork uses, just add a file `freemarker.properties` to the classpath. The supported properties are those that the freemarker Configuration object expects - see the freemarker documentation for these. These properties are used for both the `freemarker` result type, and the webwork provided `FreemarkerServlet`.

```
default_encoding=ISO-8859-1
template_update_delay=5
locale=no_NO
```

Using webwork UI tags - or any JSP Tag Library

Freemarker has builtin support for using any JSP taglib. You can use JSP taglibs in FreeMarker even if

- your servlet container has no support for JSP, or
- you didn't specify the taglib in your `web.xml` - note how in the example below we refer to the taglib by its webapp-absolute URL, so no configuration in `web.xml` is needed.

```
<#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>
```

NOTE : numeric properties for tags MUST be numbers, not strings. as in the rows and cols properties above. if you use `cols="40"` you will receive an exception. Other than that, the freemarker tag container behaves as you would expect.

Using the FreemarkerServlet

The `FreemarkerServlet` provided in the `freemarker.jar` will work out of the box **however** it won't provide any webwork specific functionality such as the context variables, property resolution etc. Therefore webwork provides its own servlet to provide this integration.

Register the FreemarkerServlet in web.xml

To use freemarker as a view engine, the webwork2 FreemarkerServlet needs to be configured, and mapped to the file extension that you use for your templates.

```
<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>
```

Configure Actions to use this servlet (xwork.xml configuration)

To use the freemarker view, just use the `dispatcher` result type, and specify the location to the template file.

```
<action name="test" class="package.Test">
  <result name="success" type="dispatcher"/>/WEB-INF/views/testView.ftl</result>
</action>
```

Extending the servlet

NOTE: these docs need to be revised, since the FreemarkerServlet has changed since they were written.

Please refer to the freemarker site for details about the base freemarker servlet.

Be careful when subclassing `com.opensymphony.webwork.views.freemarker.FreemarkerServlet` when overriding

```
protected TemplateModel createModel(
  ObjectWrapper wrapper,
  ServletContext servletContext,
  HttpServletRequest request,
  HttpServletResponse response)
```

Please call `super.createModel(...)` and wrap it with a new model to avoid problems with action property resolution.

HttpHeader Result

This page last changed on Dec 20, 2005 by [mrdon](#).

A custom Result type for evaluating HTTP headers against the ValueStack.

Parameters

- **status** - the http servlet response status code that should be set on a response.
- **parse** - true by default. If set to false, the headers param will not be parsed for Ognl expressions.
- **headers** - header values.

Examples

```
<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 Dec 20, 2005 by [mrdon](#).

Generates a JasperReports report using the specified format or PDF if no format is specified.

Parameters

- **location (default)** - the location where the compiled jasper report definition is (foo.jasper), relative from current URL.
- **dataSource (required)** - the Ognl expression used to retrieve the datasource from the value stack (usually a List).
- **parse** - true by default. If set to false, the location param will not be parsed for Ognl expressions.
- **format** - the format in which the report should be generated. Valid values can be found in JasperReportConstants. If no format is specified, PDF will be used.
- **contentDisposition** - disposition (defaults to "inline", values are typically *filename="document.pdf"*).
- **documentName** - name of the document (will generate the http header Content-disposition = X; filename=X.[format]).
- **delimiter** - the delimiter used when generating CSV reports. By default, the character used is ",".
- **imageServletUrl** - name of the url that, when prefixed with the context page, can return report images.

This result follows the same rules from WebWorkResultSupport. Specifically, all parameters will be parsed if the "parse" parameter is not set to false.

Examples

```
<result name="success" type="jasper">
    <param name="location">foo.jasper</param>
    <param name="dataSource">mySource</param>
    <param name="format">CSV</param>
</result>
```

or for 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 Mar 11, 2006 by [tm_jee](#).

A result that send the content out as plain text. Usefull typically when needed to display the raw content of a JSP or Html file for example.

Parameters

- location (default) = location of the file (jsp/html) to be displayed as plain text.
- charSet (optional) = character set to be used. This character set will be used to set the response type (eg. Content-Type=text/plain; charset=UTF-8) and when reading using a Reader. Some example of charSet would be UTF-8, ISO-8859-1 etc.

Examples

```
<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 Dec 11, 2005 by [plightbo](#).

This result uses the ActionMapper provided by the ActionMapperFactory to redirect the browser to a URL that invokes the specified action and (optional) namespace. This is better than the ServletRedirectResult because it does not require you to encode the URL patterns processed by the ActionMapper in to your xwork.xml configuration files. This means you can change your URL patterns at any point and your application will still work. It is strongly recommended that if you are redirecting to another action, you use this result rather than the standard redirect result.

To pass parameters, the <param> ... </param> tag. The following parameters will not be passable because they are part of the config param for this particular result.

- actionName
- namespace
- method
- encode
- parse
- location
- prependServletContext

See examples below for an example of how request parameters could be passed in.



See [ActionMapper](#) for more details

Parameters

- **actionName (default)** - the name of the action that will be redirect to
- **namespace** - used to determine which namespace the action is in that we're redirecting to . If namespace is null, this defaults to the current namespace

Examples

```
<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>

<package name="passingRequestParameters" extends="webwork-default"
namespace="/passingRequestParameters">
    <!-- Pass parameters (reportType, width and height) -->
    <!--
        The redirect-action url generated will be :
        /genReport/generateReport.action?reportType=pie&width=100&height=100
    -->
    <action name="gatherReportInfo" class="...">
        <result name="showReportResult" type="redirect-action">
            <param name="actionName">generateReport</param>
            <param name="namespace">/genReport</param>
            <param name="reportType">pie</param>
            <param name="width">100</param>
            <param name="height">100</param>
        </result>
    </action>
</package>
```

Redirect Result

This page last changed on Dec 20, 2005 by [mrdon](#).

Calls the {@link HttpServletResponse#sendRedirect(String) sendRedirect} method to the location specified. The response is told to redirect the browser to the specified location (a new request from the client). The consequence of doing this means that the action (action instance, action errors, field errors, etc) that was just executed is lost and no longer available. This is because actions are built on a single-thread model. The only way to pass data is through the session or with web parameters (url?name=value) which can be OGNL expressions.

Parameters

- **location (default)** - the location to go to after execution.
- **parse** - true by default. If set to false, the location param will not be parsed for Ognl expressions.

This result follows the same rules from WebWorkResultSupport.

Examples

```
<result name="success" type="redirect">
  <param name="location">foo.jsp</param>
  <param name="parse">false</param>
</result>
```

Stream Result

This page last changed on Dec 21, 2005 by [mrdon](#).

A custom Result type for send raw data (via an InputStream) directly to the HttpServletResponse. Very useful for allowing users to download content.

Parameters

- **contentType** - the stream mime-type as sent to the web browser (default = `text/plain`).
- **contentLength** - the stream length in bytes (the browser displays a progress bar).
- **contentDisposition** - the content disposition header value for specifying the file name (default = `inline`, values are typically `filename="document.pdf"`).
- **inputName** - the name of the InputStream property from the chained action (default = `inputStream`).
- **bufferSize** - the size of the buffer to copy from input to output (default = 1024).

Examples

```
<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 Dec 21, 2005 by [mrdon](#).

Using the Servlet container's JspFactory, this result mocks a JSP execution environment and then displays a Velocity template that will be streamed directly to the servlet output.

Parameters

- **location (default)** - the location of the template to process.
- **parse** - true by default. If set to false, the location param will not be parsed for Ognl expressions.

This result follows the same rules from WebWorkResultSupport.

Examples

```
<result name="success" type="velocity">
  <param name="location">foo.vm</param>
</result>
```

XSL Result

This page last changed on Dec 21, 2005 by [mrdon](#).

Parameters

Examples

Tags

This page last changed on Mar 21, 2006 by [phil](#).

Webwork provides a tag library decoupled from the view technology. In this section, we describe each tag in general terms, such as the attributes it supports, what the behaviors are, etc. Most tags are supported in all template languages (see [JSP Tags](#), [Velocity Tags](#), and [FreeMarker Tags](#)), but some are currently only specific to one language. Whenever a tag doesn't have complete support for every langauge, it will be noted in the reference documents.

The types of tags can be broken in to two types: general and HTML. Besides function and responsibility, the biggest difference between the general tags and the HTML tags is the fact that the HTML tags support *templates* and *themes*. In addition to the general tag reference, we also provide examples for using these generic tags in each of the support languages.



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

General Tags

General tags are used for controlling the execution flow when your pages render. They also allow for data extraction from places other than your action or the value stack, such as [Internationalization](#), JavaBeans, and including additional URLs or action executions.

1. [Control Tags](#) provide control flow, such as *if*, *else*, and *iterator*.
2. [Data Tags](#) allow for data manipulation or creation, such as *bean*, *push*, and *i18n*.

HTML Tags

Unlike the general tags, the HTML tags do not provide much control structure or logic. Rather, they are focussed on using data, either from your action/value stack or from the [Data Tags](#), and displaying it in rich and reusable HTML. All HTML tags have a unique behavior that they are driven by *templates* and *themes*. While the general tags simply output some content directly from the tag (if there is any content to output), the HTML tags defer to a template, often grouped together as a theme, to do the actual rendering.

This unique template support allows for you to use the HTML tags to build a rich set of reusable UI components that fit your exact requirements. Please read the [Themes and Templates](#) guide for more information on this powerful feature.

1. [Themes and Templates](#): a must-read explanation of how themes and templates are uses when rendering HTML tags.
2. [Form Tags](#) provide all form-related HTML output, such as *form*, *textfield*, and *select*.
3. [Non Form Tags](#) provide all non-form-related HTML output, such as *a*, *div*, and *tabbedPanel*.

Language Specific Tag Support

WebWork strives to support whatever environment you are most comfortable working in. That is why WebWork does not require a single template language, but instead allows for almost any common language to be used and even provides hooks for new languages. By default, almost every single tag is supported in JSP, Velocity, and FreeMarker. In each of these sections, you'll find examples and techniques for applying the generic tag reference toward your specific language or template choice.



As of WebWork 2.2, FreeMarker has become the "standard" template language recommended by the WebWork team. There are many reasons for this decision, which can be found in various forums archives, but it pretty much boils down to this: FreeMarker provides a richer set of features than Velocity and is also more developer-friendly when errors occur (ie: the error reports are more accurate). JSP, while still used, is much more difficult for applications that demand a more modular approach, such as changing the templates at runtime or uploading packaged "modules" of WebWork actions and template files.

1. [JSP Tags](#)
2. [Velocity Tags](#)
3. [FreeMarker Tags](#)

Control Tags

This page last changed on Jan 07, 2006 by [tm_jee](#).

Controls tags provide the ability to manipulate collections and conditionally produce content.



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

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 Jan 09, 2007 by [pfarrell](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Component for AppendIteratorTag, which jobs is to append iterators to form an appended iterator whereby entries goes from one iterator to another after each respective iterator is exhausted of entries.

For example, if there are 3 iterator appended (each iterator has 3 entries), the following will be how the appended iterator entries will be arranged:

1. First Entry of the First Iterator
2. Second Entry of the First Iterator
3. Third Entry of the First Iterator
4. First Entry of the Second Iterator
5. Second Entry of the Second Iterator
6. Third Entry of the Second Iterator
7. First Entry of the Third Iterator
8. Second Entry of the Third Iterator
9. Third Entry of the Third Iterator

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/AppendIterator.html>. The system administrator has been notified.

Example

```
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 Jan 09, 2007 by [pfarrell](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Perform basic condition flow. 'If' tag could be used by itself or with 'Else If' Tag and/or single/multiple 'Else' Tag.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Else.html>. The system administrator has been notified.

Examples

```
<ww:if test="#">

Will Not Be Executed

>
</ww:if>
<ww:elseif test="#">

Will Be Executed

>
</ww:elseif>
<ww:else>
</ww:else>
```

elseIf

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Perform basic condition flow. 'If' tag could be used by itself or with 'Else If' Tag and/or single/multiple 'Else' Tag.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/ElseIf.html>. The system administrator has been notified.

Examples

```
<ww:if test="#">

Will Not Be Executed

</ww:if>
<ww:elseif test="#">

Will Be Executed

</ww:elseif>
<ww:else>
</ww:else>
```

generator

This page last changed on Jan 10, 2007 by [phil](#).

Description

NOTE: JSP-TAG

Generate an iterator based on the val attribute supplied.

NOTE: The generated iterator will **ALWAYS** be pushed into the top of the stack, and popped at the end of the tag.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/IteratorGeneratorTag.html>.
The system administrator has been notified.

Examples

```
Example One:  
<pre>  
Generate a simple iterator  
<ww:generator val="<%{'aaa,bbb,ccc,ddd,eee'}%">  
<ww:iterator>  
    <ww:property /><br/>  
</ww:iterator>  
</ww:generator>  
</pre>  
This generates an iterator and print it out using the iterator tag.  
  
Example Two:  
<pre>  
Generate an iterator with count attribute  
<ww:generator val="<%{'aaa,bbb,ccc,ddd,eee'}%" count="3">  
<ww:iterator>  
    <ww:property /><br/>  
</ww:iterator>  
</ww:generator>  
</pre>  
This generates an iterator, but only 3 entries will be available in the iterator generated, namely aaa, bbb and ccc respectively because count attribute is set to 3  
  
Example Three:  
<pre>  
Generate an iterator with id attribute  
<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/>  
    }  
<%>  
</pre>  
This generates an iterator and put it in the PageContext under the key as specified by the id attribute.
```

```

Example Four:
<pre>
Generate an iterator with comparator attribute
<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;
            }
        };
    }

    ...
}

</pre>
This will generate an iterator with each entries decided by the converter supplied. With
this converter, it simply add "converter-" to each entries.

```

if

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Perform basic condition flow. 'If' tag could be used by itself or with 'Else If' Tag and/or single/multiple 'Else' Tag.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/If.html>. The system administrator has been notified.

Examples

```
<ww:if test="#">

Will Not Be Executed

>
</ww:if>
<ww:elseif test="#">

Will Be Executed

>
</ww:elseif>
<ww:else>
</ww:else>
```

Links

[Why won't the 'if' tag evaluate a one char string](#)

iterator

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Iterator will iterate over a value. An iterable value can be either of: java.util.Collection, java.util.Iterator,

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/IteratorComponent.html>. The system administrator has been notified.

Examples

The following example retrieves the value of the getDays() method of the current object on the value stack and uses it to iterate over. The <ww:property> tag prints out the current value of the iterator.

```
<ww:iterator value="days">
    <p>day is: <ww:property/></p>
</ww:iterator>
```

The following example uses a Bean tag and places it into the ActionContext. The iterator tag will retrieve that object from the ActionContext and then calls its getDays() method as above. The status attribute is also used to create a IteratorStatus object, which in this example, its odd() method is used to alternate row colours:

```
<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>
<tr>
    <td><ww:property /></td>
</tr>
</table>
</p>
```

The next example will further demonstrate the use of the status attribute, using a DAO obtained from the action class through OGNL, iterating over groups and their users (in a security context). The last() method indicates if the current object is the last available in the iteration, and if not, we need to separate the users using a comma:

```
<webwork:iterator value="groupDao.groups" status="groupStatus">
    <tr class="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>
```

The next example iterates over a an action collection and passes every iterator value to another action. The trick here lies in the use of the '[0]' operator. It takes the current iterator value and passes it on to the edit action. Using the '[0]' operator has the same effect as using >ww:property />. (The latter, however, does not work from inside the param tag).

```
<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>
```

To simulate a simple loop with iterator tag, the following could be done. It does the loop 5 times.

```
<ww:iterator status="stat" value="{1,2,3,4,5}" >
    <!-- grab the index (start with 0 ... ) -->
    <ww:property value="#stat.index" />

    <!-- grab the top of the stack which should be the -->
    <!-- current iteration value (0, 1, ... 5) -->
    <ww:property value="top" />
</ww:iterator>
```

merge

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Component for MergeIteratorTag, which job is to merge iterators and successive call to the merged iterator will cause each merge iterator to have a chance to expose its element, subsequently next call will allow the next iterator to expose its element. Once the last iterator is done exposing its element, the first iterator is allowed to do so again (unless it is exhausted of entries).

Internally the task are delegated to MergeIteratorFilter

Example if there are 3 lists being merged, each list have 3 entries, the following will be the logic.

1. Display first element of the first list
2. Display first element of the second list
3. Display first element of the third list
4. Display second element of the first list
5. Display second element of the second list
6. Display second element of the third list
7. Display third element of the first list
8. Display thrid element of the second list
9. Display third element of the thrid list

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/MergeIterator.html>. The system administrator has been notified.

Examples

```
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;  
    }  
}
```

```
        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 Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

NOTE: JSP-TAG

A Tag that sorts a List using a Comparator both passed in as the tag attribute. If 'id' attribute is specified, the sorted list will be placed into the PageContext attribute using the key specified by 'id'. The sorted list will ALWAYS be pushed into the stack and popped at the end of this tag.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/SortIteratorTag.html>. The system administrator has been notified.

Examples

```
USAGE 1:  
<ww:sort comparator="myComparator" source="myList">  
    <ww:iterator>  
        <!-- do something with each sorted elements -->  
        <ww:property value="..." />  
    </ww:iterator>  
</ww:sort>  
  
USAGE 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 Jan 10, 2007 by [phil](#).

Description

NOTE: JSP-TAG

A tag that takes an iterator and outputs a subset of it. It delegates to com.opensymphony.webwork.util.SubsetIteratorFilter internally to perform the subset functionality.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/SubsetIteratorTag.html>. The system administrator has been notified.

Examples

```
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 getList() {
        return l;
    }

    public Decider getDecider() {
        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 Nov 28, 2005 by [tm_jee](#).

Data tags provide various data-related functionality. This ranges from displaying the direct result of an action, to retrieving localized values.

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 Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

This tag enables developers to call actions directly from a JSP page by specifying the action name and an optional namespace. The body content of the tag is used to render the results from the Action. Any result processor defined for this action in xwork.xml will be ignored, *unless* the executeResult parameter is specified.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/ActionComponent.html>. The system administrator has been notified.

Examples

```
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 Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Instantiates a class that conforms to the JavaBeans specification. This tag has a body which can contain a number of Param elements to set any mutator methods on that class.

If the id attribute is set on the BeanTag, it will place the instantiated bean into the stack's Context.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Bean.html>. The system administrator has been notified.

Examples

```
<-- 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>
```

This example instantiates a bean called SimpleCounter and sets the foo property (setFoo('BAR')). The SimpleCounter object is then pushed onto the Valuestack, which means that we can call its accessor methods (getFoo()) with the Property tag and get their values.

In the above example, the id has been set to a value of *counter*. This means that the SimpleCounter class will be placed into the stack's context. You can access the SimpleCounter class using WW's tag:

```
<-- jsp form -->
<ww:property value="#counter" />

<-- freemarker form -->
[ww:property value="#counter.foo"/]
```

In the property tag example, the # tells Ognl to search the context for the SimpleCounter class which has an id(key) of *counter*

debug

This page last changed on Oct 11, 2005 by [digi9ten](#).

debug

Attribute	Type	Required	Default	Description
id	string	FALSE		

i18n

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Gets a resource bundle and place it on the value stack. This allows the text tag to access messages from any bundle, and not just the bundle associated with the current action.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/I18n.html>. The system administrator has been notified.

Examples

```
<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 Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Include a servlet's output (result of servlet or a JSP page).

Note: Any additional params supplied to the included page are **not** accessible within the rendered page through the <ww:property...> tag!

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Include.html>. The system administrator has been notified.

Example

```
<-- 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>
```

```
Example one - do an include myJsp.jsp page
Example two - do an include to myJsp.jsp page with parameters param1=value1 and param2=value2
Example three - do an include to myJsp.jsp page with parameters param1=value1 and param2=value2
```

param

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

This tag can be used to parameterize other tags.

The include tag and bean tag are examples of such tags. The parameters can be added with or without a name as key. If the tag provides a name attribute the parameters are added using the {@link Component#addParameter(String, Object) addParamter} method. For unnamed parameters the Tag must implement the UnnamedParametric interface defined in this class (e.g. The TextTag does this). This tag has the following two paramters.

- name (String) - the name of the parameter
- value (Object) - the value of the parameter

Note: When you declare the param tag, the value can be defined in either a `value` attribute or as text between the start and end tag. WebWork behaves a bit different according to these two situations. This is best illustrated using an example:

```
<param name="color">blue</param> <-- (A) -->
<param name="color" value="blue"/> <-- (B) -->
```

In the first situation (A) the value would be evaluated to the stack as a `java.lang.String` object. And in situation (B) the value would be evaluated to the stack as a `java.lang.Object` object.

For more information see [WW-808](#).

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Param.html>. The system administrator has been notified.

Examples

```
<pre>
<ui:component>
  <ui:param name="key"      value="[0]" />
  <ui:param name="value"    value="[1]" />
  <ui:param name="context"  value="[2]" />
</ui:component>
</pre>
```

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 Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

- Used to get the property of a <i>value</i>, which will default to the top of the stack if none is specified.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Property.html>. The system administrator has been notified.

Examples

```
<ww:push value="myBean">
  <!-- Example 1: -->
  <ww:property value="myBeanProperty" />

  <!-- Example 2: -->
  <ww:property value="myBeanProperty" default="a default value" />
</ww:push>
```

Example 1 prints the result of myBean's `getMyBeanProperty()` method.
Example 2 prints the result of myBean's `getMyBeanProperty()` method and if it is null, print 'a default value' instead.

push

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Push value on stack for simplified usage.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Push.html>. The system administrator has been notified.

Examples

```
<ww:push value="user">
    <ww:property value="firstName" />
    <ww:property value="lastName" />
</ww:push>
```

Pushed user into the stack, and hence property tag could access user's properties (firstName, lastName etc) since user is not at the top of the stack

```
<ww:push value="myObject">
    <ww:bean name="jp.SomeBean" id="myBean"/>           ----- (1)
    <ww:param name="myParam" value="top"/>             ----- (2)
    </ww:bean>                                         ----- (3)
</ww:push>
```

when in (1), myObject is at the top of the stack
when in (2), jp.SomeBean is in the top of stack, also in stack's context with key myBean
when in (3), top will get the jp.SomeBean instance

```
<ww:push value="myObject">
    <ww:bean name="jp.SomeBean" id="myBean"/>           ---(A)
    <ww:param name="myParam" value="top.mySomeOtherValue"/> ---(B)
    </ww:bean>                                         ---(C)
</ww:push>
```

when in (A), myObject is at the top of the stack
when in (B), jp.SomeBean is at the top of the stack, also in context with key myBean
when in (C), top refers to jp.SomeBean instance. so top.mySomeOtherValue would invoke SomeBean's mySomeOtherValue() method

```
<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>
```

when in (i), myObject is at the top of the stack
when in (ii), jp.SomeBean is at the top of the stack, followed by myObject
when in (iii), [1].top will return the top of the cut of stack starting from myObject, namely myObject itself

set

This page last changed on Mar 23, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

The set tag assigns a value to a variable in a specified scope. It is useful when you wish to assign a variable to a complex expression and then simply reference that variable each time rather than the complex expression. This is useful in both cases: when the complex expression takes time (performance improvement) or is hard to read (code readability improvement).

The set tag accepts body. However the followings need to be take note when using set tag with body

- body are treated as String and will not be parsed by OGNL
- body could be scriptlet or JSP tags, the String representation of scriptlet or JSP tags will be used
- A non-empty will take precedence if there's also a value attribute present, if the body is empty, then the value attribute will be used

Parameters

Examples

```

<ww:set name="personName" value="person.name"/>
Hello, <ww:property value="#{personName}"/>. How are you?

<ww:set name="personName">
  <ww:property value="{'some string'}" />
</ww:set>
Hello, <ww:property value="#{personName}"/>. How are you?

<ww:set name="personName">
  <c:set value="${person.name}" />
</ww:set>
Hello, <ww:property value="#{personName}"/>. How are you?

```

Name	Required	Default	Type	Description
name	true		String	The name of the new variable that is assigned the value of <i>value</i>
scope	false	action	String	The scope in which to assign the variable. Can be application , session , request , page , or action .
value	false		Object/String	The value that is assigned to the variable named <i>name</i>
id	false		Object/String	id for referencing element. For UI and form tags it will be used as HTML id attribute

text

This page last changed on Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Render a I18n text message.

The message must be in a resource bundle with the same name as the action that it is associated with. In practice this means that you should create a properties file in the same package as your Java class with the same name as your class, but with .properties extension.

If the named message is not found, then the body of the tag will be used as default message. If no body is used, then the name of the message will be used.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Text.html>. The system administrator has been notified.

Examples

Accessing messages from a given bundle (the i18n Shop example bundle in the first example) and using bundle defined through `ww` in the second example.</p>

```
<!-- First Example -->
<ww:i18n name="webwork.action.test.i18n.Shop">
    <ww:text name="main.title"/>
</ww:i18n>

<!-- Second Example -->
<ww:text name="main.title" />
```

Other example

```
<ww:text name="format.money"><ww:param name="value" value="myMoneyValue" /></ww:text>
```

where

```
format.money={0,number,currency}
```

For documentation on possibilities for formatting text, see

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 Jan 10, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.



Namespace should start with slash '/', as this is what WebWork (DefaultActionMapper) expects.

Description

This tag is used to create a URL.

You can use the "param" tag inside the body to provide additional request parameters.

NOTE:

When includeParams is 'all' or 'get', the parameter defined in param tag will take precedence and will not be overriden if they exists in the parameter submitted. For example, in Example 3 below, if there is a id parameter in the url where the page this tag is included like

`http://editUser.action?id=3333&name=John` the generated url will be

`http://context>/editUser.action?id=22&name=John` cause the parameter defined in the param tag will take precedence.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/URL.html>. The system administrator has been notified.

Examples

```
<-- 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 Jul 07, 2006 by [tm_jee](#).

Description

Within the form tags, there are two classes of tags: the form tag itself, and all other tags, which make up the individual form elements. This is important as the behavior of the form tag itself is different than that of the elements enclosed within it. Before we go provide a reference for all the form tags, including the form tag itself, we must outline some general characteristics first.



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Form Tag Themes

As previously noted in [Themes and Templates](#), the HTML Tags (which includes Form Tags) are all driven by templates. Templates are grouped together to form themes. By default, WebWork provides three themes:

- simple
- xhtml, which extends simple (default)
- ajax, which extends xhtml

Remember: the xhtml theme renders out a two-column table. If you need a different layout, we highly recommend that you do *not* write your own HTML, but rather create your own theme or utilize the simple theme.

The downside of using the simple theme is that it doesn't support as many of the attributes that the other themes do. For example, the label attribute does nothing in the simple theme. Similarly, the functionality offered by the simple theme is much less than that of the xhtml and ajax themes: the automatic display of error messages is not supported.

Common Attributes

All the form tags extend the UIBean class. This base class generally common attributes, grouped in to three classes: templated-related, javascript-related, and general attributes. We won't document what these attributes do here as that is taken care of in each individual tag's reference. However, it is a good idea to familiarize yourself with the structure of the UI tags and what attributes are available for all tags.

In addition to these attributes, a special attribute exists for all form element tags: `form`(ie: `#{parameters.form}`). This represents the parameters used to render the form tag and allows you to provide interaction between your form elements and the form itself. For example, in a template you could

access the form's ID by calling \${parameters.form.id}.

Template-Related Attributes

Attribute	Theme	Data Types	Description
templateDir	n/a	String	define the template directory
theme	n/a	String	define the theme name
template	n/a	String	define the template name

Javascript-Related Attributes

Attribute	Theme	Data Types	Description
onclick	simple	String	html javascript onclick attribute
ondbclick	simple	String	html javascript ondbclick attribute
onmousedown	simple	String	html javascript onmousedown attribute
onmouseup	simple	String	html javascript onmouseup attribute
onmouseover	simple	String	html javascript onmouseover attribute
onmouseout	simple	String	html javascript onmouseout attribute
onfocus	simple	String	html javascript onfocus attribute
onblur	simple	String	html javascript onblur attribute
onkeypress	simple	String	html javascript onkeypress attribute
onkeyup	simple	String	html javascript onkeyup attribute
onkeydown	simple	String	html javascript onkeydown attribute
onselect	simple	String	html javascript onselect attribute
onchange	simple	String	html javascript onchange attribute

Tooltip Related Attributes

Attribute	Data Type	Default	Description
tooltip	String	none	Set the tooltip of this particular component
tooltipIcon	String	/webwork/static/tooltip/toolip.gif	Set the tooltip icon
tooltipAboveMousePointer	Boolean	false	Places the tooltip above

			the mousepointer. Additionally applied the tooltipOffsetY allows to set the vertical distance from the mousepointer.
tooltipBgColor	String	#e6ecff	Background color of the tooltip.
tooltipBgImg	String	none	Background image.
tooltipBorderWidth	String	1	Width of tooltip border.
tooltipBorderColor	String	#003399	Background color of the tooltip
tooltipDelay	String	500	Tooltip shows up after the specified timeout (milliseconds). A behavior similar to that of OS based tooltips.
tooltipFixCoordinateX	String	not specified	Fixes the tooltip to the X co-ordinates specified. Useful for example if combined with tooltipSticky attribute.
tooltipFixCoordinateY	String	not specified	Fixes the tooltip to the Y co-ordinates specified. Useful for example if combined with tooltipSticky attribute.
tooltipFontColor	String	#000066	Font color.
tooltipFontFace	String	arial,helvetica,sans-serif	Font face/family eg. verdana,geneva,sans-serif
tooltipFontSize	String	11px	Font size + unit eg. 30px
tooltipFontWeight	String	normal	Font weight. either normal or bold
tooltipLeftOfMousePointer	Boolean	false	Tooltip positioned on the left side of the mousepointer
tooltipOffsetX	String	12	Horizontal offset from mouse-pointer.
tooltipOffsetY	String	15	Vertical offset from mouse-pointer.
tooltipOpacity	String	100	Transparency of tooltip. Opacity is the opposite of transparency. Value must be a number between 0 (fully transparent) and 100 (opaque, no transparency). Not (yet) supported by Opera.
tooltipPadding	String	3	Inner spacing, ie. the spacing between border and content, for instance text or image(s)
tooltipShadowColor	String	#cccccc	Creates shadow with the specified color.

tooltipShadowWidth	String	5	Creates shadow with the specified width (offset).
tooltipStatic	Boolean	false	Like OS-based tooltips, the tooltip doesn't follow the movements of the mouse pointer.
tooltipSticky	Boolean	false	The tooltip stays fixed on its initial position until another tooltip is activated, or the user clicks on the document.
tooltipStayAppearTime	String	0	Specifies a time span in milliseconds after which the tooltip disappears, even if the mousepointer is still on the concerned HTML element, with value <=0 it acts as if no time span is defined
tooltipTextAlign	String	left	Aligns the text of both the title and the body of the tooltip. Either right, left or justify
tooltipTitle	String	none	title
tooltipTitleColor	String	#ffffff	Color of the title text
tooltipWidth	String	300	Width of tooltip

General Attributes

Attribute	Theme	Data Types	Description
cssClass	simple	String	define html class attribute
cssStyle	simple	String	define html style attribute
title	simple	String	define html title attribute
disabled	simple	String	define html disabled attribute
label	xhtml	String	define label of form element
labelPosition	xhtml	String	define label position of form element (top/left), default to left
requiredposition	xhtml	String	define required label position of form element (left/right), default to right
name	simple	String	Form Element's field name mapping
required	xhtml	Boolean	add * to label (true to add false otherwise)
tabIndex	simple	String	define html tabindex attribute
value	simple	Object	define value of form

Attribute	Theme	Data Types	Description
			element

When Some Attributes Don't Apply

Note that some tags don't have any templates that utilize certain attributes, either because it doesn't make sense or it isn't required. For example, the form tag, while it supports the `tabindex` attribute, none of the themes render it out. Also, as mentioned, certain themes won't utilize some attributes.

Value/Name Relationship

In many of the tags, except for the form tag, there is a unique relationship between the `name` and `value` attributes. The `name` attribute is what the form element gets named and eventually submitted as. This effectively is the expression to which you wish to bind the incoming value to. In most cases, it is a simple JavaBean property, such as "firstName". This would eventually call `setFirstName()`.

Similarly, you often wish to also display in your form elements existing data from the same JavaBean property. This time, the attribute `value` is used. A value of "`%{firstName}`" would call `getFirstName()` and display it in your form, allowing users to edit the value and re-submit it.

You could use the following code, and it would work just fine:

```
<@ww.form action="updatePerson">
    <@ww.textfield label="First name" name="firstName" value="%{firstName}" />
    ...
</@ww.form>
```

However, because the relationship between `name` and `value` is so often predictable, we automatically do this for you, allowing you to do:

```
<@ww.form action="updatePerson">
    <@ww.textfield label="First name" name="firstName" />
    ...
</@ww.form>
```

While most attributes are exposed to the underlying templates as the same key as the attribute (ie: `#{parameters.label}`), the `value` attribute is not. Instead, it can be accessed via the "nameValue" key (ie: `#{parameters.nameValue}`) to indicate that it may have been generated from the `name` attribute rather than explicitly defined in the `value` attribute.

ID Name Assignment

All form tags automatically assign an ID for you. You are free to override this ID if you wish. The ID assignment works as follows:

1. For forms, the ID is assumed to be the action name. In the previous example, the ID would be "updatePerson".
2. For form elements, the ID is assumed to be [form's ID]_[element name]

Required Attribute

The "required" attribute on many WebWork UI tags defaults to true only if you have client side validation enabled and there is a validator associated with that particular field.

Tooltip

Every Form UI component (in xhtml / css_xhtml or any others that extends of them) could have tooltip assigned to a them. The Form component's tooltip related attribute once defined will be applicable to all form UI component that is created under it unless explicitly overriden by having the Form UI component itself defined that tooltip attribute.

In Example 1, the textfield will inherit the tooltipAboveMousePointer attribute from its containing form. In other words, although it doesn't define a tooltipAboveMousePointer attribute, it will have that attribute defined as true inherited from its containing form.

In Example 2, the the textfield will inherite both the tooltipAboveMousePointer and tooltipLeftOfMousePointer attribute from its containing form but tooltipLeftOfMousePointer attribute is overriden at the textfield itself. Hence, the textfield actually will have tooltipAboveMousePointer defined as true, inherited from its containing form and tooltipLeftOfMousePointer defined as false, due to overriden at the textfield itself.

Example 3, 4 and 5 shows different way of setting the tooltipConfig attribute.

Example 3: Set tooltip config through body of param tag

Example 4: Set tooltip config through value attribute of param tag

Example 5: Set tooltip config through tooltipConfig attribute of component tag

```
<!-- Example 1: -->
<ww:form
    tooltipConfig="#{'tooltipAboveMousePointer':'true',
    'tooltipBgColor='#eeeeee'}" .... >
...
<ww:textfield label="Customer Name" tooltip="Enter the customer name" .... />
...
</ww:form>

<!-- Example 2: -->
<ww:form
    tooltipConfig="#{'tooltipAboveMousePointer':'true',
    'tooltipLeftOfMousePointer':'true'}" ... >
...
<ww:textfield label="Address"
    tooltip="Enter your address"
    tooltipConfig="#{'tooltipLeftOfMousePointer':'false'}" />
...
</ww:form>

<!-- Example 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>

<-- Example 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>

<-- Example 5: -->
<ww:textfield
    label="Customer Telephone Number"
    tooltip="Enter customer Telephone Number"
    tooltipConfig="#{'tooltipBgColor':'#cccccc',
        'tooltipFontColor':'#eeeeee',
        'tooltipAboveMousePointer':'false',
        'tooltipLeftOfMousePointer':'false'}" />

```

Form Tag Reference



It's very important to note that all tags that insert something into the valuestack (like i18n or bean tags) will remove those objects from the stack on its end tag. This means that if you instantiate a bean with the bean tag (<ww:bean name=""br.univap.fcc.sgpw.util.FormattersHelper"">) that bean will be available on the valuestack only until the </ww:bean> tag.

1. [checkbox](#) - renders a checkbox input field
2. [checkboxlist](#) - renders a list of checkboxes
3. [combobox](#) - renders a widget that fills a text box from a select
4. [datepicker](#) - renders a date selection widget using JavaScript and DOM
5. [doubleselect](#) - renders a double select widget, where the second drop down depends on the first
6. [head](#) - renders the HEAD section for specific themes, such as CSS and JavaScript imports
7. [file](#) - renders a file input field
8. [form](#) - renders an input form
9. [hidden](#) - renders a hidden form field
10. [label](#) - renders a label
11. [optiontransferselect](#) - renders an option transfer select component which is basically two select box with buttons in between allowing entries of each select to get transfer between each other.
12. [optgroup](#) - renders a optgroup tag to be used within a select tag
13. [password](#) - renders a password textfield
14. [radio](#) - renders a radio button
15. [reset](#) - renders a reset form button
16. [richtexteditor](#) - renders a rich text editor
17. [select](#) - renders a select
18. [submit](#) - renders a submit button
19. [textarea](#) - renders a textarea

20. [textfield](#) - renders a textfield
21. [token](#) - renders a hidden field to stop double-submission of containing forms
22. [updownselect](#) - renders a select component with buttons to move the elements in the select component up and down

checkbox

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Renders an HTML input element of type checkbox, populated by the specified property from the OgnlValueStack.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Checkbox.html>. The system administrator has been notified.

Examples

```
JSP:  
<ww:checkbox label="checkbox test" name="checkboxField1" value="aBoolean" fieldValue="true"/>  
  
Velocity:  
#tag( Checkbox "label=checkbox test" "name=checkboxField1" "value=aBoolean" )  
  
Resulting HTML (simple template, aBoolean == true):  
<input type="checkbox" name="checkboxField1" value="true" checked="checked" />
```

checkboxlist

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.



Important

Note that the listkey and listvalue attribute will default to "key" and "value" respectively only when the list attribute is evaluated to a Map or its descendant. Other thing else, will result in listkey and listvalue to be null and not used.

Description

Creates a series of checkboxes from a list. Setup is like <ww:select /> or <ww:radio />, but creates checkbox tags.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/CheckboxList.html>. The system administrator has been notified.

Examples

```
<ww:checkboxlist name="foo" list="bar" />
```

It is possible to select multiple checkboxes, by handling a List into the value attribute. The List passed into the value attribute could be :-

- key if the list attribute is passed in is a Map
- listKey properties if listKey attribute is used
- entry/entries passed into the list attribute

```
<ww:checkboxlist name="options"
                  list="#{'FOO':'foo','BAR':'bar','BAZ':'baz','BOO':'boo'}"
                  value="#{'FOO','BAZ'}" />

<ww:checkboxlist name="options"
                  list="#{'Foo','Bar','Baz'}"
                  value="#{'Foo','Bar'}" />

public class MyAction extends ActionSupport {
    public List<Choice> getChoices() {
        ....
    }
    ....
    public List<String> getPreSelectedChoices() {
```

```
// returns a list of Choice.getKey(), which is a String
    ...
}

public class Choice {
    public String getKey() { ... }
    public String getDisplayName() { ... }
    ...
}

<ww:checkboxlist name="myChoice"
    list="#{choices}"
    listKey="#{key}"
    listValue="#{displayName}"
    value="#{preSelectedChoices}" />
```

combobox

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

The combo box is basically an HTML INPUT of type text and HTML SELECT grouped together to give you a combo box functionality. You can place text in the INPUT control by using the SELECT control or type it in directly in the text field.

In this example, the SELECT will be populated from id=year attribute. Counter is itself an Iterator. It will span from first to last. The population is done via javascript, and requires that this tag be surrounded by a <form>.

Note that unlike the <ww:select/> tag, there is no ability to define the individual <option> tags' id attribute or content separately. Each of these is simply populated from the toString() method of the list item. Presumably this is because the select box isn't intended to actually submit useful data, but to assist the user in filling out the text field.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/ComboBox.html>. The system administrator has been notified.

Examples

```
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 Mar 19, 2007 by [dres1011](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Renders datepicker element.

Implementation was changed in WebWork 2.2 to use [jcalendar](#) instead of non locale aware tigracalendar. Check locale and format settings if you used the old widget in your applications. Be sure to include proper stylesheet as described below if you don't want the calender widget to look transparent.

Important: Be sure to set the id attributs if not used within a <ww:form /> tag, as it takes care of setting the id for you, being required to copy selected date to text input element.

Following a reference for the format parameter (copied from jcalendar documentation):

%a	abbreviated weekday name
%A	full weekday name
%b	abbreviated month name
%B	full month name
%C	century number
%d	the day of the month (00 .. 31)
%e	the day of the month (0 .. 31)
%H	hour (00 .. 23)
%I	hour (01 .. 12)
%j	day of the year (000 .. 366)
%k	hour (0 .. 23)
%l	hour (1 .. 12)
%m	month (01 .. 12)
%M	minute (00 .. 59)
%n	a newline character
%p	"PM" or "AM"
%P	"pm" or "am"
%S	second (00 .. 59)
%s	number of seconds since Epoch (since Jan 01 1970 00:00:00 UTC)
%t	a tab character
%U, %W, %V	the week number
%u	the day of the week (1 .. 7, 1 = MON)
%w	the day of the week (0 .. 6, 0 = SUN)
%Y	year without the century (00 .. 99)
%Y	year including the century (ex. 1979)
%%	a literal % character

Parameters

Examples

```
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" />
```

If you use this jscalendar based datepicker widget, you might want to use one of the standard stylesheets provided with jscalendar (all distribution stylesheets are included in webwork jar). The easiest way to do so is to place the `<ww:head/>` tag in the head of your html page, as it takes care of including calendar css. Otherwise, to manually activate the calendar-blue style, include the following in your stylesheet definition:

```
<link href="<ww:url value="/webwork/jscalendar/calendar-blue.css" />" rel="stylesheet"  
type="text/css" media="all"/>
```

Name	Required	Default	Type	Description
language	false	The language of the current Locale	String	The language to use for the widget texts and localization presets.
format	false	Dateformat specified by language preset (%Y/%m/%d for en)	String	The format to use for date field.
showstime	false	false	String	Whether time selector is to be shown. Valid values are "true", "false", "24" and "12".
singleclick	false	true	Boolean	Whether to use selected value after single or double click.
maxlength	false		Integer	HTML maxlength attribute
maxLength	false		Object/String	Deprecated. Use maxlength instead.
readonly	false	false	Boolean	Whether the input is readonly
size	false		Integer	HTML size attribute
theme	false		Object/String	The theme (other than default) to use for rendering the element
templateDir	false		Object/String	The template directory (other than default) to used to

template	false		Object/String	find the themes and hence the template. The template (other than default) to use for rendering the element
cssClass	false		Object/String	The css class to use for element
cssStyle	false		Object/String	The css style definitions for element to use
title	false		Object/String	Set the html title attribute on rendered html element
disabled	false		Object/String	Set the html disabled attribute on rendered html element
label	false		Object/String	Label expression used for rendering a element specific label
labelPosition labelposition	false false	left	Object/String Object/String	deprecated. define label position of form element (top/left)
requiredposition	false		Object/String	define required position of required form element (left right)
name	false		Object/String	The name to set for element
required	false	false	Boolean	If set to true, the rendered element will indicate that input is required
tabindex	false		Object/String	Set the html tabindex attribute on rendered html element
value	false		Object/String	Preset the value of input element.
onclick	false		Object/String	Set the html onclick attribute on rendered html element
ondblclick	false		Object/String	Set the html ondblclick attribute on rendered html element
onmousedown	false		Object/String	Set the html onmousedown attribute on

onmouseup	false	Object/String	rendered html element Set the html onmouseup attribute on rendered html element
onmouseover	false	Object/String	Set the html onmouseover attribute on rendered html element
onmousemove	false	Object/String	Set the html onmousemove attribute on rendered html element
onmouseout	false	Object/String	Set the html onmouseout attribute on rendered html element
onfocus	false	Object/String	Set the html onfocus attribute on rendered html element
onblur	false	Object/String	Set the html onblur attribute on rendered html element
onkeypress	false	Object/String	Set the html onkeypress attribute on rendered html element
onkeydown	false	Object/String	Set the html onkeydown attribute on rendered html element
onkeyup	false	Object/String	Set the html onkeyup attribute on rendered html element
onselect	false	Object/String	Set the html onselect attribute on rendered html element
onchange	false	Object/String	Set the html onchange attribute on rendered html element
accesskey	false	Object/String	Set the html accesskey attribute on rendered html element
tooltip	false	String	Set the tooltip of this

tooltipConfig	false	String	particular component Set the tooltip configuration
id	false	Object/String	id for referencing element. For UI and form tags it will be used as HTML id attribute

doubleselect

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.



Important

Note that the doublelistkey and doublelistvalue attribute will default to "key" and "value" respectively only when the doublelist attribute is evaluated to a Map or its descendant. Other thing else, will result in doublelistkey and doublelistvalue to be null and not used.

Description

Renders two HTML select elements with second one changing displayed values depending on selected entry of first one.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/DoubleSelect.html>. The system administrator has been notified.

Examples

```
<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'}" />
```

fielderror

This page last changed on Jan 08, 2007 by [phil](#).

Description

Render field errors if they exists. Specific layout depends on the particular theme.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/FieldError.html>. The system administrator has been notified.

Examples

```
<!-- example 1 -->
<ww:fielderror />

<!-- example 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>
```

Example 1: display all field errors Example 2: display field errors only for 'field1' and 'field2'

file

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Renders an HTML file input element.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/File.html>. The system administrator has been notified.

Examples

```
<ww:file name="anUploadFile" accept="text/*" />
<ww:file name="anohterUploadFILE" accept="text/html,text/plain" />
```

form

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Renders HTML an input form.

The remote form allows the form to be submitted without the page being refreshed. The results from the form can be inserted into any HTML element on the page.

NOTE: The order / logic in determining the posting url of the generated HTML form is as follows:-

1. If the action attribute is not specified, then the current request will be used to determine the posting url
2. If the action is given, WebWork will try to obtain an ActionConfig. This will be successfull if the action attribute is a valid action alias defined xwork.xml.
3. If the action is given and is not an action alias defined in xwork.xml WebWork will used the action attribute as if it is the posting url, separating the namespace from it and using UrlHelper to generate the final url.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Form.html>. The system administrator has been notified.

Examples

```
<ww:form ... />
```

Validation

There are two flavours [Client Side Validation](#), depening on the theme you are using (xhtml, ajax, etc). If you are using the [xhtml theme](#) or [css_xhtml theme](#), pure client side validation will be used. If you are using the [ajax theme](#), a special AJAX-based validation will take place. Read the [Client Side Validation](#) docs for more information.

head

This page last changed on Mar 19, 2007 by [dres1011](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Renders parts of the HEAD section for an HTML file. This is useful as some themes require certain CSS and JavaScript includes.

If, for example, your page has ajax components integrated, without having the default theme set to ajax, you might want to use the head tag with **theme="ajax"** so that the typical ajax header setup will be included in the page.

The tag also includes the option to set a custom datepicker theme if needed. See calendarcss parameter for description for details.

If you use the ajax theme you can turn a debug flag on by setting the debug parameter to true.

Parameters

Examples

```
<head>
  <title>My page</title>
  <ww:head/>
</head>
```

```
<head>
  <title>My page</title>
  <ww:head theme="ajax" calendarcss="calendar-green"/>
</head>
```

Name	Required	Default	Type	Description
calendarcss	false		Object/String	The jscalendar css theme to use" default="calendar-blue.css
debug	false		Object/String	Set to true to enable debugging mode for AJAX themes
theme	false		Object/String	The theme (other than default) to use for rendering the element
templateDir	false		Object/String	The template directory (other than default) to used to find the themes and hence the template.
template	false		Object/String	The template (other than default) to use for rendering the element
cssClass	false		Object/String	The css class to use for element
cssStyle	false		Object/String	The css style definitions for element ro use
title	false		Object/String	Set the html title attribute on rendered html element
disabled	false		Object/String	Set the html disabled attribute on rendered html element
label	false		Object/String	Label expression used for rendering a element specific label
labelPosition	false	left	Object/String	deprecated.
labelposition	false		Object/String	define label position

requiredposition	false		Object/String	of form element (top/left) define required position of required form element (left right)
name	false		Object/String	The name to set for element
required	false	false	Boolean	If set to true, the rendered element will indicate that input is required
tabindex	false		Object/String	Set the html tabindex attribute on rendered html element
value	false		Object/String	Preset the value of input element.
onclick	false		Object/String	Set the html onclick attribute on rendered html element
ondblclick	false		Object/String	Set the html ondblclick attribute on rendered html element
onmousedown	false		Object/String	Set the html onmousedown attribute on rendered html element
onmouseup	false		Object/String	Set the html onmouseup attribute on rendered html element
onmouseover	false		Object/String	Set the html onmouseover attribute on rendered html element
onmousemove	false		Object/String	Set the html onmousemove attribute on rendered html element
onmouseout	false		Object/String	Set the html onmouseout attribute on rendered html element
onfocus	false		Object/String	Set the html onfocus attribute on rendered html element

onblur	false	Object/String	Set the html onblur attribute on rendered html element
onkeypress	false	Object/String	Set the html onkeypress attribute on rendered html element
onkeydown	false	Object/String	Set the html onkeydown attribute on rendered html element
onkeyup	false	Object/String	Set the html onkeyup attribute on rendered html element
onselect	false	Object/String	Set the html onselect attribute on rendered html element
onchange	false	Object/String	Set the html onchange attribute on rendered html element
accesskey	false	Object/String	Set the html accesskey attribute on rendered html element
tooltip	false	String	Set the tooltip of this particular component
tooltipConfig	false	String	Set the tooltip configuration
id	false	Object/String	id for referencing element. For UI and form tags it will be used as HTML id attribute

hidden

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Renders an HTML input element of type hidden, populated by the specified property from the OgnlValueStack.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Hidden.html>. The system administrator has been notified.

Examples

```
<!-- example one -->
<ww:hidden name="foo" />
<!-- example two -->
<ww:hidden name="foo" value="bar" />

Example One Resulting HTML (if foo evaluates to bar):
<input type="hidden" name="foo" value="bar" />
Example Two Resulting HTML (if getBar method of the action returns 'bar')
<input type="hidden" name="foo" value="bar" />
```

label

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Renders an HTML LABEL that will allow you to output label:name combination that has the same format treatment as the rest of your UI controls.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Label.html>. The system administrator has been notified.

Examples

```
<ww:label label="%{text('user_name')} name="userName" />
```

In this example, a label is rendered. The label is retrieved from a ResourceBundle by calling ActionSupport's getText() method giving you an output of 'User Name:tm_jee'. Assuming that i18n message user_name corresponds to 'User Name' and the action's getUsername() method returns 'tm_jee'<p/>

optgroup

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Create a optgroup component which needs to resides within a select tag.



This component is to be used within a Select component.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/OptGroup.html>. The system administrator has been notified.

Examples

```
<ww:select label="My Selection"
            name="mySelection"
            value="%{'POPEYE'}"
            list="%{#{'SUPERMAN':'Superman', 'SPIDERMAN':'spiderman'}}">
    <ww:optgroup label="Adult"
                  list="%{#{'SOUTH_PARK':'South Park'}}" />
    <ww:optgroup label="Japanese"
                  list="%{#{'POKEMON':'pokemon', 'DIGIMON':'digimon', 'SAILORMOON':'Sailormoon'}}" />
</ww:select>
```

optiontransferselect

This page last changed on Jan 08, 2007 by [tm_jee](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.



Important

Note that the doublelistkey and doublelistvalue attribute will default to "key" and "value" respectively only when the doublelist attribute is evaluated to a Map or its descendant. Other thing else, will result in doublelistkey and doublelistvalue to be null and not used.

Description

Create a option transfer select component which is basically two <select ...> tag with buttons in the middle of them allowing options in each of the <select ...> to be moved between themselves. Will auto-select all its elements upon its containing form submision.

NOTE: The id and doubleId need not be supplied as they will generated provided that the optiontransferselect tag is being used in a form tag. The generated id and doubleId will be <form_id>_<optiontransferselect_doubleName> and <form_id>_<optiontransferselect_doubleName> respectively.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/OptionTransferSelect.html>.
The system administrator has been notified.

Examples

```
<!-- 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 Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Render an HTML input tag of type password.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Password.html>. The system administrator has been notified.

Examples

In this example, a password control is displayed. For the label, we are calling ActionSupport's getText() to retrieve password label from a resource bundle.<p>

```
<ww:password label="%{getText('password')}" name="password" size="10" maxlength="15" />
```

radio

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.



Important

Note that the listkey and listvalue attribute will default to "key" and "value" respectively only when the list attribute is evaluated to a Map or its descendant. Other thing else, will result in listkey and listvalue to be null and not used.

Description

Render a radio button input field.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Radio.html>. The system administrator has been notified.

Examples

In this example, a radio control is displayed with a list of genders. The gender list is built from attribute id=genders. WW calls getGenders() which will return a Map. For examples using listKey and listValue attributes, see the section select tag. The default selected one will be determined (in this case) by the getMale() method in the action class which should retun a value similar to the key of the getGenters() map if that particular gender is to be selected.

```
<ww:action name="GenderMap" id="genders" />
<ww:radio label="Gender" name="male" list="#genders.genders" />
```

reset

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Render a reset button. The reset tag is used together with the form tag to provide form resetting. The reset can have two different types of rendering:

- input: renders as html <input type="reset" ...>
- button: renders as html <button type="reset" ...>

Please note that the button type has advantages by adding the possibility to separate the submitted value from the text shown on the button face, but has issues with Microsoft Internet Explorer at least up to 6.0

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Reset.html>. The system administrator has been notified.

Examples

Example 1

```
<ww:reset value="%{'Reset'}" />
```

Example 2

```
Render an button reset:  
<ww:reset type="button" value="%{'Reset'}" label="Reset the form"/>
```

richtexteditor

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Create a Rich Text Editor based on FCK editor (www.fckeditor.net).

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/RichTextEditor.html>. The system administrator has been notified.

Examples

```
<ww:richtexteditor  
    toolbarCanCollapse="false"  
    width="700"  
    label="Description 1"  
    name="description1"  
    value="Some Content I keyed In In The Tag Itself"  
/>
```

Server Side Browsing

It is possible to have a rich text editor do server side browsing when for example the image button is clicked. To integrate this functionality with webwork, one need to defined the following action definition typically in xwork.xml

```
<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>
```

By default whenever a browse command is triggered (eg. by clicking on the 'image' button and then 'browse server' button, the url </webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?&Type=Image&Connector=connector>

The page browser.html which comes with FCK Editor will trigger the url '/webwork/richtexteditor/editor/filemanager/browser/default/connectors/jsp/connector.action' which will caused the webwork's DefaultRichtexteditorConnector to be executed. The trigerring url could be changed by altering the 'imageBrowseURL'. There 3 types of such related url, namely 'imageBrowseURL', 'linkBrowseURL' and 'flashBrowseURL'. It is recomended that the default one being used. One could change the Connector parameter instead. For example

```
/webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?  
&Type=Image&Connector=connectors/jsp/connector.action
```

could be changed to

```
/webwork/static/richtexteditor/editor/filemanager/browser/default/browser.html?  
&Type=Image&Connector=myLittlePath/myConnector.action
```

In this case the action will need to have a namespace of '/webwork/richtexteditor/editor/filemanager/browser/default/myLittlePath' and action name of 'myConnector'

By default the action method that needs to be defined in xwork.xml needs to be 'browse'. If this needs to be something else say, myBrowse, the following could be used

```
public String myBrowse() {  
    browse();  
}
```

Server Side Uploading

It is possible for the richtexteditor to do server side uploading as well. For example when clicking on the 'Image' button and then the 'Upload' tab and then selecting a file from client local machine and the clicking 'Send it to the server'. To integrate this functionality with webwork, one need to defined the following action definition typically in xwork.xml

```
<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>
```

By default whenever an upload command is triggered, a '/webwork/static/richtexteditor/editor/filemanager/upload/uploader.action?Type=Image' will be issued. This could be changed by setting the imageUploadURL attribute of the tag. When this link is issued, the webwork action will get executed. There's 3 such related upload url namely, 'imageUploadURL', 'linkUploadURL' and 'flashUploadURL'. It is recomended that the default one being used. However one could change the url, but need to include the Type parameter. For example

```
/webwork/static/richtexteditor/editor/filemanager/upload/uploader.action?Type=Image
```

could be changed to

```
/webwork/static/richtexteditor/editor/filemanager/upload/aDifferentUploader.action?Type=Image
```

In this case the action will need to have a namespace of
'/webwork/static/richtexteditor/editor/filemanager/upload' and action name of 'aDifferentUploader'

By default the action method that needs to be defined in xwork.xml needs to be 'upload'. If this needs to be something else say, myUpload, the following could be used

```
public String myUpload() {  
    upload();  
}
```

WebWork Action

AbstractRichtexteditorConnector

An abstract class to be extended in order for the Rich text editor to perform server-side browsing and uploading.

The webwork action that handles the server-side browsing and uploading needs to extends from AbstractRichtexteditorConnector.

There are four abstract methods need to be implemented, namely

```
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 method

The method that does the functionality when the richtexteditor 'browse' command is issued.

Following are the result name that gets returned depending on the actual 'browse' command.

Browse Command	Result Name
GetFolders	getFolders
GetFoldersAndFiles	getFoldersAndFiles
CreateFolder	createFolder
FileUpload	fileUpload

upload method

The method that does the functionality when the richtexteditor 'upload' command is '/webwork/richtexteditor/data/' issued.

It return a result name of 'fileUpload'.

getFolders method

Method that gets called when a 'GetFolders' command is issued by the rich text editor. This method should search the server-side and return an Folder[] that the server side has.

The folder path queried by the rich text editor is `FolderPath`. While the type of could be one of 'Image', 'Link' or 'Flash'.

getFoldersAndFiles method

Method that gets called when a 'GetFoldersAndFiles' command is issued by the rich text editor. This method should typically search the server-side for files and folders under the provided `virtualFolderPath` and return a `FoldersAndFiles` object.

The folder path queried by the richtexted editor is `virtualFolderPath`. While the type could be one of 'Image', 'Link' or 'Flash'.

createFolder method

Method that gets called when a 'CreateFolder' command is issued by the rich text editor. This method would typically create a folder in the server-side if it is allowed to do so and return the result through `CreateFolderResult` object. `CreateFolderResult` contains static methods to return the available results.

The folder path queried by the richtexted editor is `virtualFolderPath`. While the type could be one of 'Image', 'Link' or 'Flash'. The new folder name to be created is `newFolderName`.

fileUpload method

Method that gets called when a 'FileUpload' command is issued by the rich text editor. This method would typically handle the file upload and return a `FileUploadResult` object. `FileUploadResult` contains only static methods that could create the available results.

The folder path queried by the richtexted editor is `virtualFolderPath`. While the type could be one of 'Image', 'Link' or 'Flash'. The upload file name is `filename` while its content type is `contentType` and its content could be read off the `newFile` object.

Result

AbstractRichTextEditorResult

Abstract result for all Rich Text Editor results. It contains common methods that might come in handy to its subclass.

Configuration of result necessary in xwork.xml (is already there by default) are as follows:

```
<!-- 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's result, creating the appropriate result (in xml form) and write to the response stream corresponding to the the Rich Text Editor's 'GetFolders' command.

An example of the response would be as follows:

```
<?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's result, creating the appropriate result (in xml form) and write to the response stream corresponding to the Rich Text Editor's 'GetFoldersAndFiles' command

An example of the response would be as follows:

```
<?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" />
    </Folders>
</Connector>
```

```
<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's result, creating the appropriate result (in xml form) and write it out to the response stream corresponding to a 'CreateFolder' command from the Rich Text Editor.

An example of the response would be as follows:

```
<?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's result, creating the appropriate result (in javascript form) and write to the response stream corresponding to a 'FileUpload' command from the Rich Text Editor.

An example of the response would be as follows:

```
<script type="text/javascript">
  window.parent.frames['frmUpload'].OnUploadCompleted(0) ;
</script>
```

select

This page last changed on Mar 13, 2007 by [dres1011](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Render an HTML input tag of type select.

Parameters

Examples

Note: For any of the tags that use lists (select probably being the most ubiquitous), which uses the OGNL list notation (see the "months" example above), it should be noted that the map key created (in the months example, the '01', '02', etc.) is typed. '1' is a char, '01' is a String, "1" is a String. This is important since if the value returned by your "value" attribute is NOT the same type as the key in the "list" attribute, they WILL NOT MATCH, even though their String values may be equivalent. If they don't match, nothing in your list will be auto-selected.<p/>

```
<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
```

Name	Required	Default	Type	Description
emptyOption	false	false	Boolean	Whether or not to add an empty (--) option after the header option
headerKey	false		Object/String	Key for first item in list. Must not be empty! "-1" and "" is correct, "" is bad.
headerValue	false		Object/String	Value expression for first item in list
multiple	false	false	Boolean	Creates a multiple select. The tag will pre-select multiple values if the values are passed as an Array (of appropriate types) via the value attribute. Passing a Collection may work too? Haven't tested this.
size	false		Integer	Size of the element

list	true		Object/String	box (# of elements to show) Iterable source to populate from. If the list is a Map (key, value), the Map key will become the option "value" parameter and the Map value will become the option body.
listKey	false		Object/String	Property of list objects to get field value from
listValue	false		Object/String	Property of list objects to get field content from
theme	false		Object/String	The theme (other than default) to use for rendering the element
templateDir	false		Object/String	The template directory (other than default) to used to find the themes and hence the template.
template	false		Object/String	The template (other than default) to use for rendering the element
cssClass	false		Object/String	The css class to use for element
cssStyle	false		Object/String	The css style definitions for element ro use
title	false		Object/String	Set the html title attribute on rendered html element
disabled	false		Object/String	Set the html disabled attribute on rendered html element
label	false		Object/String	Label expression used for rendering a element specific label
labelPosition	false	left	Object/String	deprecated.
labelposition	false		Object/String	define label position of form element (top/left)
requiredposition	false		Object/String	define required position of required

				form element (left right)
name	false		Object/String	The name to set for element
required	false	false	Boolean	If set to true, the rendered element will indicate that input is required
tabindex	false		Object/String	Set the html tabindex attribute on rendered html element
value	false		Object/String	Preset the value of input element.
onclick	false		Object/String	Set the html onclick attribute on rendered html element
ondblclick	false		Object/String	Set the html ondblclick attribute on rendered html element
onmousedown	false		Object/String	Set the html onmousedown attribute on rendered html element
onmouseup	false		Object/String	Set the html onmouseup attribute on rendered html element
onmouseover	false		Object/String	Set the html onmouseover attribute on rendered html element
onmousemove	false		Object/String	Set the html onmousemove attribute on rendered html element
onmouseout	false		Object/String	Set the html onmouseout attribute on rendered html element
onfocus	false		Object/String	Set the html onfocus attribute on rendered html element
onblur	false		Object/String	Set the html onblur attribute on rendered html element

onkeypress	false	Object/String	Set the html onkeypress attribute on rendered html element
onkeydown	false	Object/String	Set the html onkeydown attribute on rendered html element
onkeyup	false	Object/String	Set the html onkeyup attribute on rendered html element
onselect	false	Object/String	Set the html onselect attribute on rendered html element
onchange	false	Object/String	Set the html onchange attribute on rendered html element
accesskey	false	Object/String	Set the html accesskey attribute on rendered html element
tooltip	false	String	Set the tooltip of this particular component
tooltipConfig	false	String	Set the tooltip configuration
id	false	Object/String	id for referencing element. For UI and form tags it will be used as HTML id attribute

submit

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Render a submit button. The submit tag is used together with the form tag to provide asynchronous form submissions. The submit can have three different types of rendering:

- input: renders as html <input type="submit" ...>
- image: renders as html <input type="image" ...>
- button: renders as html <button type="submit" ...>

Please note that the button type has advantages by adding the possibility to separate the submitted value from the text shown on the button face, but has issues with Microsoft Internet Explorer at least up to 6.0

THE FOLLOWING IS ONLY VALID WHEN AJAX IS CONFIGURED

- resultDivId
- notifyTopics
- onLoadJS
- preInvokeJS

The remote form has three basic modes of use, using the resultDivId, the notifyTopics, or the onLoadJS. You can mix and match any combination of them to get your desired result. All of these examples are contained in the Ajax example webapp. Lets go through some scenarios to see how you might use it:



This tag works with all themes, but has special importance when combined with the [form](#) tag in the [ajax theme](#). Please read up on the [ajax theme](#) for more information.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Submit.html>. The system administrator has been notified.

Examples

Example 1

```
<ww:submit value="%{'Submit'}" />
```

Example 2

```
Render an image submit:  
<ww:submit type="image" value="%{'Submit'}" label="Submit the form" src="submit.gif"/>
```

Example 3

```
Render an button submit:  
<ww:submit type="button" value="%{'Submit'}" label="Submit the form"/>
```

Example 4

Show the results in another div. If you want your results to be shown in a div, use the resultDivId where the id is the id of the div you want them shown in. This is an inner HTML approach. Your results get jammed into the div for you. Here is a sample of this approach:

```
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 >
```

Example 5

Notify other controls(divs) of a change. Using an pub-sub model you can notify others that your control changed and they can take the appropriate action. Most likely they will execute some action to refresh. The notifyTopics does this for you. You can have many topic names in a comma delimited list. eg: notifyTopics="newPerson, dataChanged" . Here is an example of this approach:

```
<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>
```

Example 6

Massage the results with JavaScript. Say that your result returns some happy XML and you want to

parse it and do lots of cool things with it. The way to do this is with a onLoadJS handler. Here you provide the name of a JavaScript function to be called back with the result and the event type. The only key is that you must use the variable names 'data' and 'type' when defining the callback. For example: onLoadJS="myFancyDancyFunction(data, type)". While I talked about XML in this example, your not limited to XML, the data in the callback will be exactly whats returned as your result. Here is an example of this approach:

```
<script language="JavaScript" type="text/javascript">
    function doGreatThings(data, type) {
        //Do whatever with your returned fragment...
        //Perhaps... 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="frml" 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 Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Render HTML textarea tag.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/TextArea.html>. The system administrator has been notified.

Example

```
<ww:textarea label="Comments" name="comments" cols="30" rows="8" />
```

textfield

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Render an HTML input field of type text

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/TextField.html>. The system administrator has been notified.

Examples

In this example, a text control is rendered. The label is retrieved from a ResourceBundle by calling ActionSupport's getText() method.<p>

```
<ww:textfield label="#{text('user_name')}" name="user" />
```

token

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.

Description

Stop double-submission of forms.

The token tag is used to help with the "double click" submission problem. It is needed if you are using the TokenInterceptor or the TokenSessionInterceptor. The `ww:token` tag merely places a hidden element that contains the unique token.

Parameters

An error occurred: [http://svn.opensymphony.com/svn/webwork/docs/tags TokenName.html](http://svn.opensymphony.com/svn/webwork/docs/tags	TokenName.html). The system administrator has been notified.

Examples

```
<ww:token />
```

updownselect

This page last changed on Jan 08, 2007 by [phil](#).



Please make sure you have read the [Tag Syntax](#) document and understand how tag attribute syntax works.



Important

Note that the listkey and listvalue attribute will default to "key" and "value" respectively only when the list attribute is evaluated to a Map or its descendant. Other thing else, will result in listkey and listvalue to be null and not used.

Description

Create a Select component with buttons to move the elements in the select component up and down. When the containing form is submitted, its elements will be submitted in the order they are arranged (top to bottom).

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/UpDownSelect.html>. The system administrator has been notified.

Examples

```
<!-- 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 Jan 07, 2006 by [plightbo](#).

FreeMarker tags are extensions of the generic [Tags](#) provided by WebWork. You can get started almost immediately by simply knowing the generic structure in which the tags can be accessed: <@ww.xxx> ...</@ww.xxx>, where xxx is any of the tags supported by WebWork.

Syntax

For example, in JSP you might create a form like so:

```
<ww:form action="updatePerson">
    <ww:textfield label="First name" name="firstName" />
    <ww:submit value="Update" />
</ww:form>
```

In FreeMarker the same form is built like so:

```
<@ww.form action="updatePerson">
    <@ww.textfield label="First name" name="firstName" />
    <@ww.submit value="Update" />
</@ww.form>
```

While this covers almost all you need to know for FreeMarker tags, there are a few other advanced features you should read about, specifically with how attributes and parameters work together, and how attribute types (String, List, etc) can affect the tag behavior.

Attributes and Parameters

Unlike older versions of JSP (in which the [JSP Tags](#) are based), FreeMarker allows for *dynamic attributes*, much like JSP 2.0. What this means is that you can supply attributes to the tags that the tag doesn't even support. Those attributes that cannot be applied directly to the tag object will instead be set on the tag's general **parameters** map.

For example, suppose you have the following code in JSP:

```
<ww:url value="somePage">
    <ww:param name="personId" value="${personId}" />
</ww:url>
```

In FreeMarker, you can simplify this as:

```
<@ww.url value="somePage" personId="${personId}" />
```

In addition to being able to replace cases where you might use the [param](#) tag, you can also use this functionality when building additional templates or themes for your [Form Tags](#). For example, suppose you created a "three column" theme to replace the typical two column theme (xhtml). You might want an additional parameter to display in the third column called "description". Your form can be:

```
<@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}`**.



Sometimes you may still wish to use the param tag, such as when you are nesting complex HTML within tags. The param tag has support beyond what FreeMarker can provide as inline attributes: it can take the entire body of the param tag and apply that as the *value* attribute.

Attribute Types

Remember that all tag attributes must first be set as Strings – they are then later evaluated (using [OGNL](#)) to a different type, such as List, int, or boolean. This generally works just fine, but it can be limiting when using FreeMarker which provides more advanced ways to apply attributes. Suppose the following example:

```
<@ww.select label="Foo label - ${foo}" name="${name}" list="%{{1, 2, 3}}"/>
```

What will happen here is that each attribute will be evaluated to a string as best it can. This may involve calling the `toString()` method on the internal FreeMarker objects in the hash. In this case, all objects will end up being exactly what you would expect. Then, when the tag runs, the *list* attribute will be converted from a String to a List using [OGNL](#)'s advanced collection support.

But suppose you wish to use FreeMarker's list or hash support instead? You can do this:

```
<@ww.select label="Foo label - ${foo}" name="${name}" list={1, 2, 3}/>
```

Notice that the list attribute no longer has quotes around it. Now it will come in to the tag as an object that can't easily be converted to a String. Normally, the tag would just call `toString()`, which would return "[1, 2, 3]" and be unable to be converted back to a List by OGNL. Rather than go through all this back and forth anyway, the FreeMarker tag support within WebWork will recognize collections and not pass them through the normal tag attribute, but instead set them directly in the **parameters** map, ready to be consumed by the template.

In the end, everything tends to do what you would expect, but it can help to understand the difference of when OGNL is being used and when it isn't, and how attribute types get converted.

JSP Tag Support

While WebWork provides native FreeMarker Tags, you might wish to use other third-party tags that are only available for JSP. Fortunately, FreeMarker has the ability to run JSP tags. To do so, you must include the JspSupportServlet outlined in [web.xml 2.1.x compatibility](#), as this allows the FreeMarker integration to get access to the required objects needed to emulate a JSP taglib container.

Once you've done that, you can simply add something like this in your templates:

```
<#assign cewolf=JspTaglibs[ "/WEB-INF/cewolf.tld"] />
...
<@cewold.xxx ... />
```

JSP Tags

This page last changed on Dec 11, 2005 by [plightbo](#).

JSP tags are extensions of the generic Tags provided by WebWork. You can get started almost immediately by simply knowing the generic structure in which the tags can be accessed: <ww:xxx> ... </ww:xxx>, where xxx is any of the tags supported by WebWork.

Tag Library Definition (TLD)

The JSP TLD is included in **webwork.jar**, meaning you can define the TLD as explained in the [web.xml](#) documentation. Once you've done this, you can add the following to your JSP:

```
<%@ taglib prefix="ww" uri="webwork" %>
```

Now you can use the tags, like so:

```
<ww:iterator value="people">
    <ww:property value="lastName"/>, <ww:property value="firstName"/>
</ww:iterator>
```

Non Form Tags

This page last changed on Mar 21, 2006 by [tm_jee](#).

1. [a](#) - renders an anchor (link)
2. [actionerror](#) - renders the error messages if they exists
3. [actionmessage](#) - renders the action messages if they exists
4. [component](#) - render a very customizable component allowing its template to be specified at the spot through theme, template attribute
5. [date](#) - render a date
6. [div](#) - render a div section
7. [fielderror](#) - render the field error message if they exists
8. [panel](#) - render a panel for tabbed panel
9. [table](#) - render a table
10. [tabbedpane](#) - (deprecated in favour of tabbedPanel)
11. [tabbedPanel](#) - render a tabbed panel
12. [tree](#) - render a tree widget
13. [treenode](#) - render a tree node within a [tree](#) widget

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

A tag that creates a HTML that when clicked calls a URL remote XMLHttpRequest call via the dojo framework. The result from the URL is executed as JavaScript. If a "listenTopics" is supplied, it will publish a 'click' message to that topic when the result is returned.



While this tag can be used with the [simple theme](#), [xhtml theme](#), and others, it is really designed to work best with the [ajax theme](#). We recommend reading the [ajax a template](#) documentation for more details.

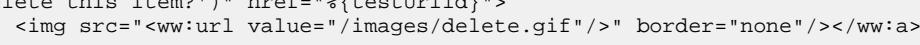
Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Anchor.html>. The system administrator has been notified.

Usage

To get started, use the [head](#) tag and the [ajax theme](#). See [ajax head template](#) for more information. Then look at the usage details for the [ajax a template](#).

If you want to use additional parameters in your ww:a the Best Practiceis to use a ww:url to create your url and then leverage this url into your ww:a tag. This is done by creating a ww:url and specifying an id attribute.. like "testUrlId" in this example. Then in the ww:a tag reference this id in the href attribute via "%{testUrlId}"

```
<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="#">
```

```
<img alt="Delete icon" data-bbox="90 780 867 809" src=""/>
```

actionerror

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

Render action errors if they exists the specific layout of the rendering depends on the theme itself.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/ActionError.html>. The system administrator has been notified.

Examples

```
<ww:actionerror />
<ww:form .... >>
    ....
</ww:form>
```

actionmessage

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

Render action messages if they exists, specific rendering layout depends on the theme itself.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/ActionMessage.html>. The system administrator has been notified.

Examples

```
<ww:actionmessage />
<ww:form .... >
    ...
</ww:form>
```

component

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

Renders an custom UI widget using the specified templates. Additional objects can be passed in to the template using the param tags.

Freemarker: Objects provided can be retrieve from within the template via `$parameters.paramname`.

Jsp: Objects provided can be retrieve from within the template via `<ww:property value="%{parameters.paramname}" />`

In the bottom JSP and Velocity samples, two parameters are being passed in to the component. From within the component, they can be accessed as:-

Freemarker: `$parameters.get('key1')` and `$parameters.get('key2')` or `$parameters.key1` and `$parameters.key2`

Jsp: `<ww:property value="%{parameters.key1}" />` and `<ww:property value="%{'parameters.key2'}" />` or `<ww:property value="%{parameters.get('key1')}" />` and `<ww:property value="%{parameters.get('key2')}" />`

Currently, your custom UI components can be written in Velocity, JSP, or Freemarker, and the correct rendering engine will be found based on file extension.

Remember: the value params will always be resolved against the OgnlValueStack so if you mean to pass a string literal to your component, make sure to wrap it in quotes i.e. `value="value1"` otherwise, the the value stack will search for an Object on the stack with a method of `getValue1()`. (now that i've written this, i'm not entirely sure this is the case. i should verify this manana)



If Jsp is used as the template, the jsp template itself must lie within the webapp itself and not the classpath. Unlike Freemarker or Velocity, JSP template could not be picked up from the classpath.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/GenericUIBean.html>. The system administrator has been notified.

Examples

```

JSP
<ww:component template="/my/custom/component.vm"/>

or

<ww:component template="/my/custom/component.vm">
  <ww:param name="key1" value="value1"/>
  <ww:param name="key2" value="value2"/>
</ww:component>

Velocity
#wwcomponent( "template=/my/custom/component.vm" )

or

#wwcomponent( "template=/my/custom/component.vm" )
  #wwparam( "name=key1" "value=value1" )
  #wwparam( "name=key2" "value=value2" )
#end

Freemarker
<@ww.component template="/my/custom/component.ftl" />

or

<@ww.component template="/my/custom/component.ftl">
  <@ww.param name="key1" value="${'value1'}" />
  <@ww.param name="key2" value="${'value2'}" />
</@ww.component>

```

date

This page last changed on Mar 19, 2007 by [dres1011](#).

Description

Format Date object in different ways.

The date tag will allow you to format a Date in a quick and easy way. You can specify a **custom format** (eg. "dd/MM/yyyy hh:mm"), you can generate **easy readable notations** (like "in 2 hours, 14 minutes"), or you can just fall back on a **predefined format** with key 'webwork.date.format' in your properties file.

If that key is not defined, it will finally fall back to the default DateFormat.MEDIUM formatting.

Note: If the requested Date object isn't found on the stack, a blank will be returned.

Configurable attributes are :-

- name
- nice
- format

Following how the date component will work, depending on the value of nice attribute (which by default is false) and the format attribute.

Condition 1: With nice attribute as true

i18n key	default
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}

Condition 2: With nice attribute as false and format attribute is specified eg. dd/MM/yyyy

In this case the format attribute will be used.

Condition 3: With nice attribute as false and no format attribute is specified

i18n key	default
----------	---------

webwork.date.format

if one is not found DateFormat.MEDIUM format will be used

Parameters

Examples

```
<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" />
```

Name	Required	Default	Type	Description
format	false		Object/String	Date or DateTime format pattern
nice	false	false	Boolean	Whether to print out the date nicely
name	true		String	The date value to format
id	false		Object/String	id for referencing element. For UI and form tags it will be used as HTML id attribute

div

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

The div tag is primarily an AJAX tag, providing a remote call from the current page to update a section of content without having to refresh the entire page.

It creates a HTML <DIV /> that obtains it's content via a remote XMLHttpRequest call via the dojo framework.

If a "listenTopics" is supplied, it will listen to that topic and refresh it's content when any message is received.



While this tag can be used with the [simple theme](#), [xhtml theme](#), and others, it is really designed to work best with the [ajax theme](#). We recommend reading the [ajax div template](#) documentation for more details.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Div.html>. The system administrator has been notified.

Usage

To get started, use the [head](#) tag and the [ajax theme](#). See [ajax head template](#) for more information. Then look at the usage details for the [ajax div template](#).



Handy Hint

I have found that the target of the div must return good html. Basically a html and a body block that are happily closed.

panel

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

Render a panel for tabbedPanel.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Panel.html>. The system administrator has been notified.

Examples

The following is an example of a tabbedpanel and panel tag utilizing local and remote content.

If you are looking for the "nifty" rounded corner look, there is additional configuration. This assumes that the background color of the tabs is white. If you are using a different color, please modify the parameter in the Rounded() method.

```
<link rel="stylesheet" type="text/css" href="">
<link rel="stylesheet" type="text/css" href="">
<link rel="stylesheet" type="text/css" href="" media="print">
<script type="text/javascript" src=""></script>
<script type="text/javascript">
    dojo.event.connect(window, "onload", function() {
        if (!NiftyCheck())
            return;
        Rounded("li.tab_selected", "top", "white", "transparent", "border #fffffS");
        Rounded("li.tab_unselected", "top", "white", "transparent", "border #fffffS");
        // "white" needs to be replaced with the background color
    });
</script>
```

tabbedpane

This page last changed on Dec 03, 2005 by [tm_jee](#).



Deprecated

This tag has been deprecated in favour of TabbedPanel.

tabbedpane

Attribute	Type	Required	Default	Description
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 Jan 09, 2007 by [pfarrell](#).

Description

The tabbedpanel widget is primarily an AJAX component, where each tab can either be local content or remote content (refreshed each time the user selects that tab).



This tag only works with the [ajax theme](#). Be sure to read up on the theme before using this tag.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/TabbedPanel.html>. The system administrator has been notified.

Examples

The following is an example of a tabbedpanel and panel tag utilizing local and remote content.

An error occurred:

<http://svn.opensymphony.com/svn/webwork/webapps/showcase/src/webapp/ajax/tabbedpanel/example3.jsp>.
The system administrator has been notified.

If you are looking for the "nifty" rounded corner look, there is additional configuration. This assumes that the background color of the tabs is white. If you are using a different color, please modify the parameter in the Rounded() method.

```
<link rel="stylesheet" type="text/css" href="">
<link rel="stylesheet" type="text/css" href="">
<link rel="stylesheet" type="text/css" href="" media="print">
<script type="text/javascript" src=""></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 Jan 09, 2007 by [pfarrell](#).

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/WebTable.html>. The system administrator has been notified.

tree

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

Renders a tree widget with AJAX support.

The id attribute is normally specified, such that it could be looked up using javascript if necessary.

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/Tree.html>. The system administrator has been notified.

Examples

```
&lt;-- statically --> <ww:tree id="..." label="..."> <ww:treenode id="..." label="..." /> <ww:treenode id="..." label="..."> <ww:treenode id="..." label="..." /> <ww:treenode id="..." label="..." />  
&lt;/ww:treenode> <ww:treenode id="..." label="..." /> </ww:tree>  
  
<-- dynamically --> <ww:tree id="..." rootNode="..." nodeIdProperty="..." nodeTitleProperty="..." childCollectionProperty="..." />
```

treenode

This page last changed on Jan 09, 2007 by [pfarrell](#).

Description

Renders a tree node within a tree widget with AJAX support.

Either of the following combinations should be used depending on if the tree is to be constructed dynamically or statically.

Dynamically

- id - id of this tree node
- title - label to be displayed for this tree node

Statically

- rootNode - the parent node of which this tree is derived from
- nodeIdProperty - property to obtain this current tree node's id
- nodeTitleProperty - property to obtain this current tree node's title
- childCollectionProperty - property that returns this current tree node's children

Parameters

An error occurred: <http://svn.opensymphony.com/svn/webwork/docs/tags/TreeNode.html>. The system administrator has been notified.

Examples

```
&lt;-- statically --> <ww:tree id="..." label="..."> <ww:treenode id="..." label="..." /> <ww:treenode id="..." label="..."> <ww:treenode id="..." label="..." /> <ww:treenode id="..." label="..." />  
&lt;/ww:treenode> <ww:treenode id="..." label="..." /> </ww:tree>
```

```
<-- dynamically --> <ww:tree id="..." rootNode="..." nodeIdProperty="..." nodeTitleProperty="..." childCollectionProperty="..." />
```

Tag Syntax

This page last changed on Dec 11, 2005 by [plightbo](#).

The tag syntax in WebWork is extremely easy to understand. To quickly get started, all you need to know is that all attributes are applied as Strings initially. They are then parsed for the syntax `%{ ... }`, and anything in between the braces is evaluated against the value stack.



Upgrade note!

The tag syntax was not always this easy – if you are upgrading from WebWork 2.1.7 or previous versions, you may wish to read about the [Alt Syntax](#).

Like most things in life, it turns out that this isn't quite *that* simple. Specifically, there are actually three rules to be aware of:

1. All *String* attribute types are *parsed* for the `%{ ... }` characters.
2. All *non-String* attribute types are **not** parsed, but instead evaluated directly as an [OGNL](#) expression
3. The exception to rule #2 is that if the *non-String* attribute starts with `%{` and ends with `}`, those characters are cut off before evaluating the expression.

The best way to understand these rules is by looking at some examples.



We recognize that these rules can be confusing. Generally, you should not need to know them at all, as 99.9% of the time everything will "just work". However, as we see in the examples, there are some tricky situations that require understanding of these rules. In future versions of WebWork, will be trying to make the tag syntax even simpler

Some Examples

The most basic example explaining how the tag syntax works is as follows. This example shows off rule #1 only:

```
<ww:textfield label="%{getText( \"state.label\" )}" name="state"/>
```

In this example, the label is dynamically evaluated and set to the outcome of the [OGNL](#) expression `getText("state.label")`, which will in turn invoke the [Internationalization](#) system are retrieve the value of the i18n key `state.label`. The name, being a String attribute, is simply set to the string `state`.

The next example shows off rule #2:

```
<ww:select label="%{getText( \"state.label\" )}" name="state" multiple="true"/>
```

While this looks very similar to the last example, the key thing to recognize is that the `multiple` attribute is of type *Boolean*, which means it falls under rule #2. Generally you won't even notice this, because `true` as an [OGNL](#) expression evaluated to true, which is what you want.

Now let's suppose we want to extend this example to show off rule #3 by making the multiple attribute dynamic:

```
<ww:select label="${getText("state.label")}" name="state" multiple="${allowMultiple}" />
```

Because the attribute is of type *Boolean* and starts and ends with the correct characters from rule #3, it is reduced to the expression *allowMultiple*, which is evaluated against the value stack, returning a true or false value, just like in the previous example.

There is one trick example to keep an eye on, however. For example, the following is probably **incorrect**:

```
<ww:textfield label="${getText("state.label")}" name="state" value="CA" />
```

This example will only work if the expression *CA* can result in something, meaning that your action has a method *getCA()*, which is probably not what you expected. This is because the *value* attribute is of type *Object* and therefore rule #2 applies. If the desire is to set a static String as the initial value, you would need to supply an [OGNL](#) expression that returns a String. For example, this is the **correct** way to do it:

```
<ww:textfield label="${getText("state.label")}" name="state" value="${'CA'}" />
```

While you could set the *value* attribute as just "CA", we recommend the parsed expressions so that, in the future when WebWork supports parsed attributes for all types, your code will still work.

Alt Syntax

This page last changed on Jan 07, 2006 by [plightbo](#).

The `altSyntax` is an option that can be defined in [webwork.properties](#). By default it is set to true and it is **strongly** recommend you do not change that unless you are upgrading from WebWork 2.1.7 or previous versions.



Migration tip

You can also turn on the altSyntax on a per-page basis by using the `set` tag. Simply set the name `useAltSyntax` to the value `true`. From this point on, all tags will use the altSyntax for the rest of the request.

The altSyntax changes the behavior of how tags are interpreted. Instead of evaluating each tag parameter against the value stack and needing single quotes to mark string literals, only marked expressions are evaluated.

Example:

the following code uses the [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>
```

this is somewhat counter intuitive to normal HTML tag behaviour, and you get loads of single quotes. Now the same example in 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>
```

Only expressions enclosed with `%{}` are evaluated. The code is shorter and clearer, very similar to JSTL EL usage. Quoting problems, eg. with javascript function calls, are avoided.

In order to fully understand why this option exists and what the differences are, it is best to get a bit of history about WebWork.



If you are *not* upgrading from WebWork 2.1.7 or previous versions and you don't care about the history of WebWork's evolution, you can skip this section. See the [Tag Syntax](#) section for more information on the standard tag syntax support

History

In WebWork 2.1.4, the altSyntax option was introduced. The book, *WebWork in Action*, while based around WebWork 2.1.7, was entirely written with the assumption that the altSyntax was enabled. As of WebWork 2.2, the altSyntax is turned on by default and eventually the old syntax will no longer be supported and will be removed from the code.

Themes and Templates

This page last changed on Jan 07, 2006 by [plightbo](#).

At the core of the HTML [Tags](#) provided by WebWork are **themes** and **templates**. We'll first start off with a simple definition of three key terms:

- **tag** - a small piece of code executed from within [JSP](#), [FreeMarker](#), or [Velocity](#).
- **template** - a bit of code, usually written in [FreeMarker](#), that can be rendered by certain tags (HTML tags)
- **theme** - a collection of *templates* packaged together to provide common functionality

Tags are covered in the [Tags](#) section, so we won't discuss them much here other than their relationship to themes and templates already stated. Instead, we'll focus on several important topics:

- [Template Loading](#) - how templates are loaded by WebWork
- [Selecting Themes](#) - how you can pick a theme when writing your results
- [Extending Themes](#) - how to create your own themes based on already existing themes
- Themes - a detailed overview of each theme included with WebWork
 - [simple theme](#) - a minimal theme with know additional "bells and whistles"
 - [xhtml theme](#) - the default theme that uses common HTML practices
 - [css_xhtml theme](#) - the [xhtml theme](#) re-implemented using strictly CSS for layout
 - [ajax theme](#) - a theme based on the [xhtml theme](#) that provides advanced AJAX features

ajax theme

This page last changed on Jan 09, 2007 by [pfarrell](#).

The ajax theme extends the [xhtml theme](#), providing AJAX features on top of everything provided by its parent theme. This theme uses two popular AJAX/JavaScript libraries: Dojo and DWR. These AJAX features are:

- [AJAX Client Side Validation](#)
- Remote [form](#) submission support (works with the [submit](#) tag as well)
- An advanced [div](#) template that provides dynamic reloading of partial HTML
- An advanced [a](#) template that provides the ability to load and evaluate JavaScript remotely
- An AJAX-only [tabbedPanel](#) implementation
- A rich pub-sub event model

Browser Compatibility

AJAX (as a technology) uses a browser side scripting component that varies between browsers (and sometimes versions). To hide those differences from the developer, we utilize the dojo toolkit (<http://www.dojotoolkit.org>). The following browsers are supported by dojo, and any UI's created with the AJAX theme should act the same for all those browsers listed below:

- IE 5.5+
- FF 1.0+
- Latest Safari (on up-to-date OS versions)
- Latest Opera
- Latest Konqueror

Extending the XHTML Theme

The wrapping behavior provided by this theme is almost exactly like that provided by the [xhtml theme](#). The only difference is that the **controlheader.ftl** template is slightly different:

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/ajax/controlheader.ftl>. The system administrator has been notified.

This provides for [AJAX Client Side Validation](#) by checking if the *validate* attribute is set to true. If it is, on each **onblur** event for HTML [Tags](#), a validation request is made. Some people don't like the onblur behavior and would rather a more advanced timer (say, 200ms) be kicked off after every keystroke. You can override this template and provide that type of behavior if you would like.

Special Notes

While most of the templates in this theme are self explanatory, there are some templates that should be called out and explained in detail:



Especially with this theme, it is strongly recommended you use the [head](#) tag. See the [ajax head template](#) for more information. Without it, you most likely not have AJAX support set up properly.

- [ajax head template](#)
- [ajax div template](#)
- [ajax a template](#)

In addition to these templates, it is important to make yourself familiar with the [ajax event system](#) provided by WebWork and Dojo.

ajax a template

This page last changed on Jan 09, 2006 by [roughley](#).

The ajax a template is used to make asynchronous calls to the server when the user clicks on the a href link. It is useful when you need to communicate information back to the application from the UI, without requiring the entire page to be re-rendered. An example would be removing an item from a list.

The *preInvokeJS* attribute is used to determine whether the URL specified should be called or not, and must contain Javascript that returns *true* or *false*. If you want to call a JavaScript function, use the format *preInvokeJS='yourMethodName(data,type)'*. An example would be to show a confirm dialog to the user to double check whether they want to remove a user from a list.

Remember: the content returned by the *href* attribute must be JavaScript. That JavaScript will then be evaluated within the webpage. If you only wish to publish an event to the topic specified, then simply return no result (or NONE) from your action and utilize the *notifyTopics* attribute to specific the topic names.

For an example of the interaction between the [div](#) tag and the [a](#) tag using the topic pub/sub model, see the examples in the [ajax div template](#).

ajax div template

This page last changed on Jul 13, 2006 by [tm_jee](#).

The ajax [div](#) template provides a much more interesting div rendering option than the other themes do. Rather than simply rendering a <div> tag, this template relies on advanced AJAX features provided by the [Dojo Toolkit](#). While the [div](#) tag could be used outside of the [ajax theme](#), it is usually not very useful. See the [div](#) tag for more information on what features are provided.

Features

The remote div has a few features, some of which can be combined with the [a](#) tag and the [ajax a template](#). These uses are:

- Retrieve remote data
- Initialize the div with content before the remote data is retrieved
- Display appropriate error and loading messages
- Refresh data on a timed cycle
- Listen for events and refresh data
- JavaScript control support

Retrieve Remote Data

The simplest way to use the div tag is to provide an *href* attribute. For example:

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239"/>
```

What this does after the HTML page is completely loaded, the specified URL will be retrieved asynchronously in the browser. The entire contents returned by that URL will be injected in to the div.

Initializing the Div

Because the remote data isn't loaded immediately, it is sometimes useful to have some placeholder content that exists before the remote data is retrieved. The content is essentially just the body of the div element. For example:

```
<ww:div theme="ajax" id="weather" href="http://www.weather.com/weather?zip=97239">
  Placeholder...
</ww:div>
```

If you wish to load more complex initial data, you can use the [action](#) tag and the *executeResult* attribute:

```
<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 and Error Messages

If you'd like to display special messages when the data is being retrieved or when the data cannot be retrieved, you can use the *errorText* and *loadingText* attributes:

```

<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>

```

Refresh Timers

Another feature this div template provides is the ability to refresh data on a timed basis. Using the *updateFreq* and the *delay* attributes, you can specify how often the timer goes off and when the timer starts (times in milliseconds). For example, the following will update every minute after a two second delay:

```

<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>

```

Listening for Events

The [a tag](#) (specifically the [ajax a template](#)) and the div tag support an [ajax event system](#), providing the ability to broadcast events to topics. You can specify the **topics** to listen to using a comma separated list in the *listenTopics* attribute. What this means is that when a topic is published, usually through the [ajax a template](#), the URL specified in the *href* attribute will be re-requested.

```

<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 occurred">Refresh</ww:a>

```

JavaScript Support

There are also javascript functions to refresh the content and stop/start the refreshing of the component. For the remote div with the component id "remotediv1":

To start refreshing use the javascript:

```
remotediv1.start();
```

To stop refreshing use the javascript:

```
remotediv1.stop();
```

To refresh the content use the javascript:

```
remotediv1.bind();
```

JavaScript Examples:

To further illustrate these concepts here is an example. Say you want to change the url of a div at runtime via javascript. Here is what you need to do:

What you will need to do is add a JS function that listens to a JS event that publishes the id from the select box that was selected. It will modify the URL for the div (adding the id so the correct data is obtained) and then bind() the AJAX div so it refreshes.

```
<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(this, "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 Jan 09, 2006 by [roughley](#).

As you may have seen with the [ajax div template](#) and [ajax a template](#), WebWork and Dojo provide a nice way to subscribe and notify of topics from within the browser. A benefit of using Dojo as the basis of many of these components is being able to loosely couple UI components. There are two attributes of importance: *listenTopics* and *notifyTopics*.

- If a component has a *notifyTopics* attribute, then after the processing has been completed a message will be published to the topic names supplied as a value (comma delimited).
- If a component has a *listenTopics* attribute, then when a message is published to the topic names supplied as a value (comma delimited), the component will perform custom tag-specific logic (i.e a DIV tag will re-fresh its content).

As well as this, you can publish and subscribe to topic names with javascript code. To publish to the *topic_name* topic:

```
dojo.event.topic.publish("topic_name", "content");
```

The *topic_name* attribute is required, the *content* attribute is not and most elements are triggered without having this attribute. See the [ajax div template](#) for an example of this type of interaction.

To subscribe to the *topic_name* topic:

```
function doSomethingWithEvent(data) {  
    ...  
}  
  
dojo.event.topic.getTopic("topic_name").subscribe(null, "doSomethingWithEvent");
```

The *subscribe* method takes 2 parameters, the first is the JavaScript object variable (or null if the function is not from an object) and the second is the name of the function to call when an event is received on the topic.

ajax head template

This page last changed on Jan 09, 2007 by [pfarrell](#).

The ajax [head](#) template builds upon the [xhtml head template](#) by providing additional JavaScript includes for the [Dojo Toolkit](#), which is used by the [ajax a template](#), [ajax div template](#), and the [ajax tabbedPanel template](#). It is required to use this tag, `<ww:head theme="ajax" />`, in your HTML `<head>` block if you wish to use AJAX feature. The contents of **head.ftl** are:

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/ajax/head.ftl>. The system administrator has been notified.



If you are having trouble getting the AJAX theme to work, you should include the above JavaScript in your page manually, changing "isDebug: false" to "isDebug: true". This will log out debugging information directly to the screen.

Note that Dojo is configured to use the same character encoding specified in [webwork.properties](#), typically UTF-8. For a simple example of how to use the [head](#) tag with the AJAX theme, simply do the following in your HTML:

An error occurred:

<http://svn.opensymphony.com/svn/webwork/webapps/showcase/src/webapp/ajax/commonInclude.jsp>.
The system administrator has been notified.

ajax submit template

This page last changed on Jan 09, 2006 by [plightbo](#).

TODO: Describe the Ajax Submit template

ajax tabbedPanel template

This page last changed on Jan 09, 2006 by [plightbo](#).

TODO: Describe the Ajax TabbedPanel template

css_xhtml theme

This page last changed on Jan 10, 2007 by [phil](#).

The css_xhtml theme provides all the basics that the [simple theme](#) provides, plus these additional features:

- Standard two-column CSS-based layout, using <div> for the HTML [Tags](#) ([form](#), [textfield](#), [select](#), etc)
- Labels for each of the HTML [Tags](#), placed according to the CSS stylesheet
- [Validation](#) error reporting
- [Pure JavaScript Client Side Validation](#) using 100% JavaScript on the browser

Wrapping the Simple Theme

The css_xhtml theme uses the **wrapping** technique mentioned in [Extending Themes](#), also done by the [xhtml theme](#). As such, it is important to understand how the HTML tags are wrapped by a standard header and footer. For example, take a look at the [textfield](#) template, [text.ftl](#):

An error occurred: http://svn.opensymphony.com/svn/webwork/src/java/template/css_xhtml/text.ftl.
The system administrator has been notified.

As you can see, the **controlheader.ftl** and **controlfooter.ftl** templates are wrapped around the simple template.

CSS XHTML Theme Header

Now let's look at the controlheader.ftl:

An error occurred:

http://svn.opensymphony.com/svn/webwork/src/java/template/css_xhtml/controlheader.ftl. The system administrator has been notified.

The header used by the HTML tags in the css_xhtml theme is somewhat complex. Unlike the [xhtml theme](#), this theme does not support the *labelposition* attribute. Instead, your CSS rules can define how the layout is done.

Also note that the **fieldErrors**, usually caused by [Validation](#), are displayed in a div block before the element is displayed.

CSS XHTML Theme Footer

And the controlfooter.ftl contents:

An error occurred:

http://svn.opensymphony.com/svn/webwork/src/java/template/css_xhtml/controlfooter.ftl. The system administrator has been notified.

Special Notes

While most of the templates in this theme are self explanatory, there are some templates that should be called out and explained in detail:

- [css_xhtml head template](#)
- [css_xhtml form template](#)

[css_xhtml form template](#)

This page last changed on Jan 08, 2006 by [plightbo](#).

The [css_xhtml form](#) template is almost exactly like the [xhtml form template](#), including support for [Pure JavaScript Client Side Validation](#). The only difference is that instead of printing out an opening and closing <table> element, there are no elements. Instead, the CSS rules for the individual HTML tags are assumed to handle all display logic. However, as noted, client side validation is still supported.

css_xhtml head template

This page last changed on Jan 10, 2007 by [phil](#).

The [css_xhtml head](#) template is very similar to the [xhtml head template](#). The only difference is that CSS that is included is specifically designed to provide the layout for the [css_xhtml theme](#). The contents of **head.ftl** are:

An error occurred: http://svn.opensymphony.com/svn/webwork/src/java/template/css_xhtml/head.ftl.
The system administrator has been notified.

The contents of **styles.css** are:

An error occurred: http://svn.opensymphony.com/svn/webwork/src/java/template/css_xhtml/styles.css.
The system administrator has been notified.

Extending Themes

This page last changed on Jan 08, 2006 by [plightbo](#).

Sometimes you may want to simply override a template (see [Template Loading](#)) or create an alternative template in an existing theme. However, other times you might want to create your own entire theme, especially if you are planning to build a rich set of unique and reusable templates for your organization.

There are three ways to create new themes:

- Create a new theme from scratch (**hard!**)
- Extend an existing theme
- Wrap an existing theme

We don't ever recommend creating a new theme from scratch. Rather, we believe that the [simple theme](#) provides enough of the basics for you to either extend or wrap, thereby creating your own unique theme. No matter which method you choose (often you end up using a bit of both methods, as they aren't mutually exclusive), we **strongly** recommend you unpack the WebWork jar and look at the source templates for all the provided themes. This will give you a good understanding of how to make your own templates and themes.

Wrapping an Existing Theme

Taking a look at the [xhtml theme](#), we can see that the templates there make extensive use of a **wrapping** technique. For example, a template might look like:

```
<#include "/${parameters.templateDir}/xhtml/controlheader.ftl" />
<#include "/${parameters.templateDir}/simple/xxx.ftl" />
<#include "/${parameters.templateDir}/xhtml/controlfooter.ftl" />
```

This template is simply wrapping the [simple theme](#)'s existing template with a header and a footer. This is a great way to add additional behavior around the basic HTML elements provided by the [simple theme](#).

Extending an Existing Theme

The theme infrastructure provided by WebWork also allows themes to **extend** an existing theme. What this means is that a theme may contain a **theme.properties** with a **parent** entry that contains the name of the theme that you would like to extend. For example, the [ajax theme](#) extends the [xhtml theme](#) in this way.

By extending a theme, you are not required to implement every single template that the [Tags](#) use. Rather, you only need to implement templates that you wish to override. All other templates will be loaded from the parent template.

Selecting Themes

This page last changed on Jan 07, 2006 by [plightbo](#).

Themes can be selected using several different rules, in this order:

1. The *theme* attribute on the specific tag
2. The *theme* attribute on a tag's surrounding [form](#) tag
3. The page-scoped attribute named *theme*
4. The request-scoped attribute named *theme*
5. The session-scoped attribute named *theme*
6. The application-scoped attribute named *theme*
7. The **webwork.ui.theme** property in [webwork.properties](#) (defaults to *xhtml*)

A few important concepts come from this order:

- You can override an entire form's theme by only changing the *theme* attribute for the forum. This makes it easy to use the [ajax theme](#) in just a few select places.
- You can change the theme for a user's session. This might be useful if users can customize their look and feel.
- If you want to change the theme for the entire application, adjust your [webwork.properties](#).

simple theme

This page last changed on Jan 08, 2006 by [plightbo](#).

The simple theme provides the "bare bones" HTML element support and is considered the low end of the structure and can be used to build additional functionality or behavior (see [Extending Themes](#) for more information). For example, the [textfield](#) tag renders the HTML <input/> tag without any additional labels, validation error reporting, or anything else. If you require additional behavior, see the [xhtml theme](#).



The [xhtml theme](#) and [css_xhtml theme](#) both directly extend this theme. Look to them for examples of how to utilize the basics provided by this theme.

While most of the templates in this theme are self explanatory, there are some templates that should be called out and explained in detail:

- [simple head template](#)

simple head template

This page last changed on Mar 19, 2007 by [dres1011](#).

The [simple theme head](#) template only does one thing: it prints out an HTML <link/> to the CSS required for the [datepicker](#) tag to render properly.

The source of the simple head.ftl template is:

```
<if parameters.calendarcss?exists>
<link rel="stylesheet" href="<@ww.url
value='/webwork/jscalendar/${parameters.calendarcss?html}' encode='false'
includeParams='none' />" type="text/css"/>
</if>
```

Template Loading

This page last changed on Jan 07, 2006 by [plightbo](#).

Unless you have some advanced or unusual requirements, template loading is very straight forward. The simple description is this:

Templates, by default, are always [FreeMarker](#) templates and are loaded the same way other FreeMarker templates (such as your results) are: by first searching the web application and then the classpath.

Template and Themes

Templates are loaded based on the theme (see [Selecting Themes](#)) and the template directory. The template directory is defined by the **webwork.ui.templateDir** property in [webwork.properties](#) (defaults to *template*). This means, by default, if a tag is using the **ajax** theme, the following two locations will be searched (in this order):

1. In the web app: /template/ajax/template.ftl
2. In the classpath: /template/ajax/template.ftl

Overriding Templates

Because most of the templates you will need are included in the WebWork jar (the classpath), you may find a few situations where you need to override a particular template to provide behavior that is unique to your application. For example, you might wish to change how [select](#) tags render. Rather than creating a brand new template and changing every tag to use that template, you can override the built in **select.ftl** template by copying the file from in the jar over to a new **/template/xhtml/select.ftl** directory.

Altering Template Loading Behavior

Sometimes it may be important to load templates from elsewhere other than the classpath and web app. For example, you might wish to load templates from the file system or a URL. This is useful for not only [HTML Tags](#), but also for any result your write.

You can do this by consulting the [FreeMarker](#) documentation and extending the FreeMarkerManager.

Alternative Template Engines

WebWork provides for template rendering engines other than [FreeMarker](#). We almost never recommend using anything other than FreeMarker templates, if only because WebWork provides templates already written in FreeMarker and not in JSP or Velocity. However, some users, especially advanced users of older

versions of WebWork (pre-2.2) may find it necessary to use a different template engine.



Alternative template engines are for advanced users. Please use with care!

WebWork supports three template engines, which can be controlled by the **webwork.ui.templateSuffix** in [webwork.properties](#):

- **ftl (default)** - [FreeMarker](#)-based template engine
- **vm** - [Velocity](#)-based template engine
- **jsp** - [JSP](#)-based template engine

If you choose to use *vm* or *jsp*, you must implement your own templates and themes entirely, which is a large amount of work.



Just because your views aren't written in FreeMarker doesn't mean you can't use the FreeMarker template engine. Again, we **strongly** discourage you from writing your own templates from scratch. Rather, we recommend you learn a small bit of FreeMarker and extend existing templates

xhtml theme

This page last changed on Jan 09, 2007 by [pfarrell](#).

The xhtml theme is the default theme in WebWork. It provides all the basics that the [simple theme](#) provides, plus these additional features:

- Standard two-column table layout for the HTML [Tags](#) ([form](#), [textfield](#), [select](#), etc)
- Labels for each of the HTML [Tags](#) on the left hand side (or top, depending on the `labelposition` attribute)
- [Validation](#) error reporting
- [Pure JavaScript Client Side Validation](#) using 100% JavaScript on the browser

Wrapping the Simple Theme

The xhtml theme uses the **wrapping** technique mentioned in [Extending Themes](#). As such, it is important to understand how the HTML tags are wrapped by a standard header and footer. For example, take a look at the [textfield](#) template, `text.ftl`:

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/text.ftl>. The system administrator has been notified.

As you can see, the `controlheader.ftl` and `controlfooter.ftl` templates are wrapped around the simple template. In case you are wondering, the reason the controlheader.ftl is referred using `#{parameters.theme}` is to assist with code re-use for the [ajax theme](#). For now, assume that the xhtml theme is used there as well.

XHTML Theme Header

Now let's look at the controlheader.ftl and controlheader-core.ftl (again, these are split up for easy re-use with the [ajax theme](#)) contents:

An error occurred:

<http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/controlheader.ftl>. The system administrator has been notified. An error occurred:

<http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/controlheader-core.ftl>. The system administrator has been notified.

The header used by the HTML tags in the xhtml theme is somewhat complex. However, if you look closely you will see that the logic produces two behaviors: either a two-column format or a two-row format. Generally the two-column approach is what you want, so that is the default option. However, you can use the two-row approach by changing the `labelposition` parameter to "top".

Also note that the **fieldErrors**, usually caused by [Validation](#), are printed out as a row above the HTML form element. Some people prefer these errors elsewhere, such as in a third column. If you wish to place these elsewhere, overriding the headers is easy, allowing you to continue to use the other features provided by this theme. See [Template Loading](#) for more information on how to do this.

XHTML Theme Footer

And the controlfooter.ftl contents:

An error occurred:

<http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/controlfooter.ftl>. The system administrator has been notified.

The important thing to note here is that the table cell and row is closed. Before that, however, note that a special *after* parameter is checked for. While this isn't an official attribute supported by any of the [Tags](#), if you are using [FreeMarker Tags](#), [Velocity Tags](#), or the [param](#) tag in any template language, you can add an *after* parameter to place any content you like after the [simple theme](#) template renders. This makes it easier to fine-tune your HTML forms as you please.

Special Notes

While most of the templates in this theme are self explanatory, there are some templates that should be called out and explained in detail:

- [xhtml head template](#)
- [xhtml form template](#)

xhtml form template

This page last changed on Jan 10, 2007 by [phil](#).

The xhtml form template sets up the wrapping table around all the other [xhtml theme](#) form elements. In addition to creating this wrapping table, the opening and closing templates also, if the *validate* parameter is set to true, enable [Pure JavaScript Client Side Validation](#). See the **form.ftl** contents:

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/form.ftl>. The system administrator has been notified.

The closing template, **form-close.ftl**:

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/form-close.ftl>.
The system administrator has been notified.

xhtml head template

This page last changed on Jan 10, 2007 by [phil](#).

The xhtml [head](#) template extends the [simple head template](#) and provides an additional CSS that helps render the [xhtml theme](#) form elements. The contents of **head.ftl** are:

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/head.ftl>. The system administrator has been notified.

The contents of **styles.css** are:

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/styles.css>. The system administrator has been notified.

Velocity Tags

This page last changed on Jan 07, 2006 by [plightbo](#).

Velocity tags are extensions of the generic [Tags](#) provided by WebWork. You can get started almost immediately by simply knowing the structure in which the tags can be accessed: **#wwxxx (...) ... #end**, where **xxx** is any of the [Tags](#) supported by WebWork.



As of WebWork 2.2, the use of Velocity for templating internal WebWork UI Tags have been removed. The new and legacy 2.2 UI tags have been implemented in FreeMarker. However, support for the Velocity result type as well as the Velocity directives shown below will continue to be supported and developed. By default, WebWork's TemplateEngine will call the FreeMarker UI Tags using the Velocity syntax noted below (through Velocity directives). See [Template Loading](#) for more information.

Syntax

For example, in JSP you might create a form like so:

```
<ww:form action="updatePerson">
    <ww:textfield label="First name" name="firstName" />
    <ww:submit value="Update" />
</ww:form>
```

In Velocity the same form is built like so:

```
#wwform ("action=updatePerson")
    #wwtextfield ("label=First name" "name=firstName")
    #wwsubmit ("value=Update")
#end
```

Block and Inline Tags

You may notice that some tags require an `#end` statement, while others do not. Due to a limitation in Velocity, tags must declare if they are a *block* or *inline* tag up front. As such, by default all tags are *inline* except for a few key ones, such as the [form](#) tag. We **strongly** encourage you to look at FreeMarker, which provides much better flexibility in this area as well as others.

Type Conversion

This page last changed on Jan 03, 2007 by [phil](#).

WebWork has one of the most advanced type conversion abilities in any web-based framework in any Java language. Generally, you don't need to do anything to take advantage of it, other than name your HTML inputs (form elements and other GET/POST parameters) names that are valid [OGNL](#) expressions.

A Simple Example

Type conversion is great for situations where you need to turn a String in to a more complex object. Because the web is type-agnostic (everything is a string in HTTP), WebWork's type conversion features are very useful. For instance, if you were prompting a user to enter in coordinates in the form of a string (such as "3, 22"), you could have WebWork do the conversion both from String to Point and from Point to String.

Using this "point" example, if your action (or another compound object in which you are setting properties on) has a corresponding `ClassName-conversion.properties` file, WebWork will use the configured type converters for conversion to and from strings. So turning "3, 22" in to new `Point(3, 22)` is done by merely adding the following entry to **ClassName-conversion.properties** (Note that the `PointConverter` should impl the `ognl.TypeConverter` interface):

```
point = com.acme.PointConverter
```

Your type converter should be sure to check what class type it is being requested to convert. Because it is used for both to and from strings, you will need to split the conversion method in to two parts: one that turns Strings in to Points, and one that turns Points in to Strings.

After this is done, you can now reference your point (using `<ww:property value="post"/>` in JSP or `${point}` in FreeMarker) and it will be printed as "3, 22" again. As such, if you submit this back to an action, it will be converted back to a Point once again.

In some situations you may wish to apply a type converter globally. This can be done by editing the file **xwork-conversion.properties** in the root of your class path (typically WEB-INF/classes) and providing a property in the form of the class name of the object you wish to convert on the left hand side and the class name of the type converter on the right hand side. For example, providing a type converter for all Point objects would mean adding the following entry:

```
com.acme.Point = com.acme.PointConverter
```



Type conversion should not be used as a substitute for i18n. It is not recommended to use this feature to print out properly formatted dates. Rather, you should use the i18n features of WebWork (and consult the JavaDocs for JDK's `MessageFormat` object) to see how a properly formatted date should be displayed.

WebWork ships with a helper base class that makes converting to and from Strings very easy. The class is **com.opensymphony.webwork.util.WebWorkTypeConverter**. This class makes it very easy for you to write type converters that handle converting objects to Strings as well as from Strings. From the JavaDocs for this class:

Base class for type converters used in WebWork. This class provides two abstract methods that are used to convert both to and from strings – the critical functionality that is core to WebWork's type coversion system.

Type converters do not have to use this class. It is merely a helper base class, although it is recommended that you use this class as it provides the common type conversion contract required for all web-based type conversion.

There's a hook (fall back method) called `performFallbackConversion` of which could be used to perform some fallback conversion if `convertValue` method of this failed. By default it just ask its super class (Ognl's `DefaultTypeConverter`) to do the conversion.

To allow WebWork to recognize that a converison error has occurred, throw an `XWorkException` or preferable a `TypeConversionException`.

Built in Type Conversion Support

WebWork will automatically handle the most common type conversion for you. This includes support for converting to and from Strings for each of the following:

- String
- boolean / Boolean
- char / Character
- int / Integer, float / Float, long / Long, double / Double
- dates - uses the SHORT format for the Locale associated with the current request
- arrays - assuming the individual strings can be converted to the individual items
- collections - if not object type can be determined, it is assumed to be a String and a new `ArrayList` is created

Note that with arrays the type conversion will defer to the type of the array elements and try to convert each item individually. As with any other type conversion, if the conversion can't be performed the standard type conversion error reporting is used to indicate a problem occured while processing the type conversion.

Relationship to Parameter Names

The best way to take advantage of WebWork's type conversion is to utilize complete objects (ideally your domain objects directly), 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. Some tips for

achieving this are:

- Use complex OGNL expressions - WebWork will automatically take care of creating the actual objects for you.
- Use JavaBeans! WebWork can only create objects for you if your objects obey the JavaBean specification and provide no-arg constructions, as well as getters and setters where appropriate.
- Remember that `person.name` will call `getPerson().setName()`, but if you are expecting WebWork to create the Person object for you, **a `setPerson()` must also exist**.
- For lists and maps, use index notation, such as `people[0].name` or `friends['patrick'].name`. Often these HTML form elements are being rendered inside a loop, so you can use the iterator tag's status attribute if you're using [JSP Tags](#) or the `${foo_index}` special property if you're using [FreeMarker Tags](#).
- For multiple select boxes, you obviously can't name each individual item using index notation. Instead, name your element simply `people.name` and WebWork will understand that it should create a new Person object for each selected item and set its name accordingly.

Creating a Type Converter

To create a type converter one would need to extends WebWorkTypeConverter.

```
public class MyConverter extends WebWorkTypeConverter {  
    public Object convertFromString(Map context, String[] values, Class toClass) {  
        ....  
    }  
  
    public String convertToString(Map context, Object o) {  
        ....  
    }  
}
```



To allow WebWork to recognize that a conversion error has occurred, the converter class needs to throw XWorkException or preferably TypeConversionException.

Advanced Type Conversion

WebWork also has some very advanced, yet easy-to-use, type conversion features. Null property handling will automatically create objects where null references are found. Collection and map support provides intelligent null handling and type conversion for Java Collections. Type conversion error handling provides an easy way to distinguish the difference between an input validation problem from an input type conversion problem.

Null Property Handling

Provided that the key `#CREATE_NULL_OBJECTS` is in the action context with a value of true (this key is set only during the execution of the `com.opensymphony.xwork.interceptor.ParametersInterceptor`), OGNL expressions that have caused a `NullPointerException` will be temporarily stopped for evaluation while the system automatically tries to solve the null references by automatically creating the object.

The following rules are used when handling null references:

- If the property is declared *exactly* as a Collection or List, then an ArrayList shall be returned and assigned to the null references.
- If the property is declared as a Map, then a HashMap will be returned and assigned to the null references.
- If the null property is a simple bean with a no-arg constructor, it will simply be created using the {@link ObjectFactory#buildBean(java.lang.Class, java.util.Map)} method.

For example, if a form element has a text field named **person.name** and the expression *person* evaluates to null, then this class will be invoked. Because the *person* expression evaluates to a *Person* class, a new Person is created and assigned to the null reference. Finally, the name is set on that object and the overall effect is that the system automatically created a Person object for you, set it by calling *setPerson()* and then finally called *getPerson().setName()* as you would typically expect.

Collection and Map Support

WebWork supports ways to determine the object type found in collections. This is done via an *ObjectTypeDeterminer*. The default implementation is provided. The JavaDocs explain how map and colelciton support is determined in the DefaultObjectTypeDeterminer:

This ObjectTypeDeterminer looks at the **Class-conversion.properties** for entries that indicated what objects are contained within Maps and Collections. For Collections, such as Lists, the element is specified using the pattern **Element_xxx**, where xxx is the field name of the collection property in your action or object. For Maps, both the key and the value may be specified by using the pattern **Key_xxx** and **Element_xxx**, respectively.

From WebWork 2.1.x, the **Collection_xxx** format is still supported and honored, although it is deprecated and will be removed eventually.

Additionally, you can create your own custom ObjectTypeDeterminer by implementing the ObjectTypeDeterminer interface. There is also an optional ObjectTypeDeterminer that utilizes Java 5 generics. See the [J2SE 5 Support](#) page for more information.

Indexing a collection by a property of that collection

It is also possible using webwork to get a unique element of a collection, by passing the value of a given property of that element. By default, 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. Here is an example with the following two classes:

```

    /**
     * @return a Collection of Foo objects
     */
    public Collection getFooCollection()
    {
        return foo;
    }

```

```

    /**
     * @return a unique identifier
     */
    public Long getId()
    {
        return id;
    }

```

Then put **KeyProperty_fooCollection=id** in my MyAction-conversion.properties file. This would allow to the use of **fooCollection(someIdValue)** to get the Foo object with value *someIdValue* in the Set fooCollection. For example, **fooCollection(22)** would return the Foo object in the fooCollection collection whose id property value was 22.

This is useful, because it ties a collection element directly to its unique identifier and therefore does not force you to use an index and thus allows you to edit the elements of a collection associated to a bean without any additional code. For example, parameter name **fooCollection(22).name** and value **Phil** would set name the Foo object in the fooCollection collection whose id property value was 22 to be Phil.

Webwork automatically converts the type of the parameter sent in to the type of the key property using type conversion.

Unlike Map and List element properties, if fooCollection(22) does not exist it will not be created. To do that, use the notation **fooCollection.makeNew[index]** where index is an integer 0, 1, and so on. Thus, parameter value pairs **fooCollection.makeNew[0]=Phil** and **fooCollection.makeNew[1]=John** would add two new Foo objects to fooCollection one with name property value Phil and the other with name property value Bar. Note, however, that in the case of a Set, the equals and hashCode methods should be defined such that they don't only include the id property. That will cause one element of the null id propertied Foos to be removed from the Set.

An advanced example for indexed Lists and Maps

Here is the model bean used within the list.

The KeyProperty for this bean is the id attribute.

```

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 + '\'' +
            '}';
    }
}

```

The action has a beanList attribute initialized with an empty ArrayList.

```

public 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;
    }
}

```

These conversion.properties tell the TypeConverter to use MyBean instances as elements of the List.

```

KeyProperty(beanList=id
Element(beanList=MyBean
CreateIfNull(beanList=true

```

When submitting this via a form, the (id) value is used as KeyProperty for the MyBean instances in the

beanList.

Notice the () notation! Do not use [] notation, this is for Maps only!

The value for name will be set to the MyBean instance with this special id.

The List does no longer have null values added for unavailable id values.

This avoids the risk of OutOfMemoryErrors!

```
<ww:iterator value="beanList" id="bean">
    <ww:textfield name="beanList(${bean.id}).name" />
</ww:iterator>
```



It would be nice to know that [] could be applied to both Map and List as well eg.

```
<ww:property value="myList[0].name" />
```

would access the 'name' property of the first element of 'myList' property (that return a List)

```
<ww:property value="myMap['myKey'].name" />
```

would access the 'name' property of the value in Map accessible through 'myMap' property which has a key of 'myKey'.

Java 5 Enumeration

What we talk about

One of Java 5's improvements is providing enumeration facility.

Up to now, there existed no enumerations. The only way to simulate was the so-called int Enum pattern:

```
public static final int SEASON_WINTER = 0;
public static final int SEASON_SPRING = 1;
public static final int SEASON_SUMMER = 2;
public static final int SEASON_FALL = 3;
```

Java 5.0 now provides the following construct:

```
public static enum Season { WINTER, SPRING, SUMMER, FALL };
```

Implementing Java 5 Enumeration Type Conversion

1. myAction-conversion.properties

- Place a myAction-conversion.properties-file in the path of your Action.
- Add the following entry to the properties-file:

```
nameOfYourField=fullyClassifiedNameOfYourConverter
```

2. myAction.java

- Your action contains the *enumeration*:

```
public enum Criticality {DEBUG, INFO, WARNING, ERROR, FATAL}
```

- Your action contains the *private field*:

```
private myEnum myFieldForEnum;
```

- Your action contains *getters and setters* for your field:

```
public myEnum getCriticality() {
    return myFieldForEnum;
}

public void setCriticality(myEnum myFieldForEnum) {
    this.myFieldForEnum= myFieldForEnum;
}
```

3. EnumTypeConverter

Your enumeration type converter looks like this:

```
/**
 * EnumTypeConverter
 * This class converts java 5 enums to String and from String[] to enum.
 * @author Tamara Cattivelli
 * @version 1.0
 * @since <pre>01.06.2006</pre>
 */
import ognl.DefaultTypeConverter;
import java.util.Map;

public class EnumTypeConverter extends DefaultTypeConverter {

    /**
     *
     */
    private static final Map<String, Object> map = new HashMap<String, Object>(10);
    static {
        map.put("DEBUG", DEBUG);
        map.put("INFO", INFO);
        map.put("WARNING", WARNING);
        map.put("ERROR", ERROR);
        map.put("FATAL", FATAL);
    }
}
```

```

        * Converts the given object to a given type. How this is to be done is
        * implemented in toClass.
        * The OGNL context, o and toClass are given.
        * This method should be able to handle conversion in general without
        * any context or object specified.
        * @param context - OGNL context under which the conversion is being done
        * @param o - the object to be converted
        * @param toClass - the class that contains the code to convert to enumeration
        * @return Converted value of type declared in toClass or
        * TypeConverter.NoConversionPossible to indicate that the conversion was not possible.
        */
    public Object convertValue(Map context, Object o, Class toClass) {
        if (o instanceof String[]) {
            return convertFromString(((String[]) o)[0], toClass);
        } else if (o instanceof String) {
            return convertFromString((String) o, toClass);
        }

        return super.convertValue(context, o, toClass);
    }

    /**
     * Converts one or more String values to the specified class.
     * @param value - the String values to be converted, such as those submitted from an
     HTML form
     * @param toClass - the class to convert to
     * @return the converted object
     * @see
com.opensymphony.webwork.util.WebWorkTypeConverter#convertFromString(java.util.Map, String[], Class)
     */
    public Object convertFromString(String value, Class toClass) {
        return Enum.valueOf(toClass, value);
    }
}

```

4. JSP

In your jsp you can access an enumeration value just normal by using the known <ww:property>-Tag:

```
<ww:property value="myField"/>
```

For any further questions contact me, Tamara Cattivelli, at [aramati@web.de]

Type Conversion Error Handling

Any error that occurs during type conversion may or may not wish to be reported. For example, reporting that the input "abc" could not be converted to a number might be important. On the other hand, reporting that an empty string, "", cannot be converted to a number might not be important - especially in a web environment where it is hard to distinguish between a user not entering a value vs. entering a blank value.

By default, all conversion errors are reported using the generic i18n key **xwork.default.invalid.fieldvalue**, which you can override (the default text is *Invalid field value for field*

"xxx", where xxx is the field name) in your global i18n resource bundle.

However, sometimes you may wish to override this message on a per-field basis. You can do this by adding an i18n key associated with just your action (Action.properties) using the pattern **invalid.fieldvalue.xxx**, where xxx is the field name.

It is important to know that none of these errors are actually reported directly. Rather, they are added to a map called *conversionErrors* in the ActionContext. There are several ways this map can then be accessed and the errors can be reported accordingly.

There are two ways the error reporting can occur:

1. globally, using the [Conversion Error Interceptor](#)
2. on a per-field basis, using the [conversion validator](#)

By default, the conversion interceptor is included in [webwork-default.xml](#) in the default stack, so if you don't want conversion errors reporting globally, you'll need to change the interceptor stack and add additional validation rules.

Validation

This page last changed on Jan 10, 2007 by [phil](#).

WebWork relies on XWork validation framework to enable the application of input validation rules to your Actions before they are executed. This section only provides the bare minimum to get you started and focuses on WebWork's extension of the XWork validators to support client-side validation.



There is also an option for Client Side (Javascript and/or AJAX) based validation, please see [Client Side Validation](#) for more information.

Examples

1. [Basic Validation](#)
2. [Client Validation](#)
3. [AJAX Validation](#)
4. [Using Field Validators](#)
5. [Using Non Field Validators](#)
6. [Using Visitor Field Validator](#)

Validators available in WebWork



Note
When using a Field Validator, Field Validator Syntax is ALWAYS preferable than using the Plain Validator Syntax as it facilitates grouping of field-validators according to fields. This is very handy especially if a field needs to have many field-validators which is almost always the case.
Examples: `validatortypes`

1. [required validator](#)
2. [requiredstring validator](#)
3. [int validator](#)
4. [date validator](#)
5. [expression validator](#)
6. [fieldexpression validator](#)
7. [email validator](#)
8. [url validator](#)
9. [visitor validator](#)
10. [conversion validator](#)
11. [stringlength validator](#)
12. [regex validator](#)
13. [collection validator](#)

Registering Validators

Validation rules are handled by validators, which must be registered with the ValidatorFactory (using the registerValidator method). The simplest way to do so is to add a file name validators.xml in the root of the classpath (/WEB-INF/classes) that declares all the validators you intend to use.

This list declares all the validators that comes with WebWork.

```
<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"/>
    <validator name="collection"
    class="com.opensymphony.xwork.validator.validators.CollectionFieldValidator" />
</validators>
```



Note

validators.xml if being defined should be available in the classpath. However this is not necessary, if no custom validator is needed. Webwork will automatically picked up a predefined sets of validators defined in com/opensymphony/xwork/validator/validators/default.xml packaged together in xwork jar file that comes with webwork distribution. See ValidatorFactory static block for details.



Warning

If custom validator is being defined and a validators.xml is created and place in the classpath, do remember to copy all the other pre-defined validators that is needed into the validators.xml as if not they will not be registered. Once a validators.xml is detected in the classpath, the default one (com/opensymphony/xwork/validator/validators/default.xml) will not be loaded. It is only loaded

when a custom validators.xml cannot be found in the classpath. Be careful.

Turning on Validation

The default validationWorkflowStack already includes this.

All that is required to enable validation for an Action is to put the ValidationInterceptor in the interceptor refs of the action (see xwork.xml) like so:

```
<interceptor name="validator" class="com.opensymphony.xwork.validator.ValidationInterceptor"/>
```

Note: The default **validationWorkflowStack** already includes this.

Validator Scopes

Field validators, as the name indicate, act on single fields accessible through an action. A validator, in contrast, is more generic and can do validations in the full action context, involving more than one field (or even no field at all) in validation rule. Most validations can be defined on per field basis. This should be preferred over non-field validation wherever possible, as field validator messages are bound to the related field and will be presented next to the corresponding input element in the respecting view.

Non-field validators only add action level messages. Non-field validators are mostly domain specific and therefore offer custom implementations. The most important standard non-field validator provided by XWork/WebWork is ExpressionValidator.

Notes

Non-field validators takes precedence over field validators regardless of the order they are defined in *.validation.xml. If a non-field validator is short-circuited, it will causes its non-field validator to not being executed. See validation framework documentation for more info.

Defining Validation Rules

Validation rules can be specified:

1. Per Action class: in a file named ActionName-validation.xml
2. Per Action alias: in a file named ActionName-alias-validation.xml
3. Inheritance hierarchy and interfaces implemented by Action class: WebWork searches up the inheritance tree of the action to find default validations for parent classes of the Action and interfaces implemented

Here is an example for SimpleAction-validation.xml:

```
<!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 lt bar </param>
        <message>Foo must be greater than Bar. Foo = ${foo}, Bar = ${bar}.</message>
    </validator>
</validators>
```

Here we can see the configuration of validators for the SimpleAction class. Validators (and fieldValidators) must have a type attribute, which refers to a name of an Validator registered with the ValidatorFactory as above. Validator elements may also have <param> elements with name and value attributes to set arbitrary parameters into the Validator instance. See below for discussion of the message element.

Each Validator or Field-Validator element must define one message element inside the validator element body. The message element has 1 attributes, key which is not required. The body of the message tag is taken as the default message which should be added to the Action if the validator fails. Key gives a message key to look up in the Action's ResourceBundles using getText() from LocaleAware if the Action implements that interface (as ActionSupport does). This provides for Localized messages based on the Locale of the user making the request (or whatever Locale you've set into the LocaleAware Action). After either retrieving the message from the ResourceBundle using the Key value, or using the Default message, the current Validator is pushed onto the ValueStack, then the message is parsed for \${...} sections which are replaced with the evaluated value of the string between the \${ and }. This allows you to parameterize your messages with values from the Validator, the Action, or both.

If the validator fails, the validator is pushed onto the ValueStack and the message - either the default or the locale-specific one if the key attribute is defined (and such a message exists) - is parsed for \${...} sections which are replaced with the evaluated value of the string between the \${ and }. This allows you to parameterize your messages with values from the validator, the Action, or both.

NOTE: Since validation rules are in an XML file, you must make sure you escape special characters. For example, notice that in the expression validator rule above we use ">" instead of ">". Consult a resource on XML for the full list of characters that must be escaped. The most commonly used characters that must be escaped are: & (use &), > (use >), and < (use <).

Here is an example of a parameterized message:

This will pull the min and max parameters from the IntRangeFieldValidator and the value of bar from the Action.

```
bar must be between ${min} and ${max}, current value is ${bar}.
```

To define validation rules for an Action, create a file named ActionName-validation.xml in the same package as the Action. You may also create alias-specific validation rules which add to the default validation rules defined in ActionName-validation.xml by creating another file in the same directory named ActionName-aliasName-validation.xml. In both cases, ActionName is the name of the Action class, and aliasName is the name of the Action alias defined in the xwork.xml configuration for the Action.

The framework will also search up the inheritance tree of the Action to find validation rules for directly implemented interfaces and parent classes of the Action. This is particularly powerful when combined with ModelDriven Actions and the VisitorFieldValidator. Here's an example of how validation rules are discovered. Given the following class structure:

- interface Animal;
- interface Quadraped extends Animal;
- class AnimalImpl implements Animal;
- class QuadrapedImpl extends AnimalImpl implements Quadraped;
- class Dog extends QuadrapedImpl;

The framework method will look for the following config files if Dog is to be validated:

- Animal
- Animal-aliasname
- AnimalImpl
- AnimalImpl-aliasname
- Quadraped
- Quadraped-aliasname
- QuadrapedImpl
- QuadrapedImpl-aliasname
- Dog
- Dog-aliasname

While this process is similar to what the XW:Localization framework does when finding messages, there are some subtle differences. The most important difference is that validation rules are discovered from the parent downwards.

NOTE: Child's *-validation.xml will override parent's *-validation.xml according to the class hierarchy defined above.

Validator Flavour

The validators supplied by the Xwork distribution (and any validators you might write yourself) come in two different flavors:

1. Plain Validators / Non-Field validators
2. FieldValidators

Plain Validators (such as the ExpressionValidator) perform validation checks that are not inherently tied to a single specified field. When you declare a plain Validator in your -validation.xml file you do not associate afieldname attribute with it. (You should avoid using plain Validators within the syntax described below.)

FieldValidators (such as the EmailValidator) are designed to perform validation checks on a single field. They require that you specify a fieldname attribute in your -validation.xml file. There are two different (but equivalent) XML syntaxes you can use to declare FieldValidators (see " vs. syntax" below).

There are two places where the differences between the two validator flavors are important to keep in mind:

1. when choosing the xml syntax used for declaring a validator (either or)
2. when using the short-circuit capability

NOTE: Note that you do not declare what "flavor" of validator you are using in your -validation.xml file, you just declare the name of the validator to use and WebWork will know whether it's a "plain Validator" or a "FieldValidator" by looking at the validation class that the validator's programmer chose to

implement.

Non-Field Validator Vs Field-Validator

There are two ways you can define validators in your -validation.xml file:

1. <validator>
2. <field-validator>

Keep the following in mind when using either syntax:

Non-Field-Validator The <validator> element allows you to declare both types of validators (either a plain Validator a field-specific FieldValidator).

```
<!-- Declaring a plain Validator using the <validator> syntax: -->  
<validator type="expression">  
    <param name="expression">foo gt bar</param>  
    <message>foo must be greater 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>
```

field-validator The <field-validator> elements are basically the same as the <validator> elements except that they inherit the fieldName attribute from the enclosing <field> element. FieldValidators defined within a <field-validator> element will have their fieldName automatically filled with the value of the parent <field> element's fieldName attribute. The reason for this structure is to conveniently group the validators for a particular field under one element, otherwise the fieldName attribute would have to be repeated, over and over, for each individual <validator>.

HINT: It is always better to define field-validator inside a <field> tag instead of using a <validator> tag and supplying fieldName as its param as the xml code itself is clearer (grouping of field is clearer)

NOTE: Note that you should only use FieldValidators (not plain Validators) within a block. A plain Validator inside a <field> will not be allowed and would generate error when parsing the xml, as it is not allowed in the defined dtd (xwork-validator-1.0.2.dtd)

Declaring a FieldValidator using the `<field-validator>` syntax:

```
<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>
```

The choice is yours. It's perfectly legal to only use elements without the elements and set the `fieldName` attribute for each of them. The following are effectively equal:

```
<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>
```

Short-Circuiting Validator

Beginning with XWork 1.0.1 (bundled with WebWork 2.1), it is possible to short-circuit a stack of validators. Here is another sample config file containing validation rules from the Xwork test cases: Notice that some of the `<field-validator>` and `<validator>` elements have the `short-circuit` attribute set to 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>
</validators>
```

```

        </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>

```

short-circuiting and Validator flavors

Plain validator takes precedence over field-validator. They get validated first in the order they are defined and then the field-validator in the order they are defined. Failure of a particular validator marked as short-circuit will prevent the evaluation of subsequent validators and an error (action error or field error depending on the type of validator) will be added to the ValidationContext of the object being validated.

In the example above, the actual execution of validator would be as follows:

1. Plain Validator 1
2. Plain Validator 2
3. Field Validators for email field
4. Field Validators for email2 field

Since Field Validator 2 is short-circuited, if its validation failed, it will cause Field validators for email field and Field validators for email2 field to not be validated as well.

Usefull Information: More complicated validation should probably be done in the validate() method on the action itself (assuming the action implements Validatable interface which ActionSupport already does).

A plain Validator (non FieldValidator) that gets short-circuited will completely break out of the validation stack ### no other validators will be evaluated and plain validator takes precedence over field validator meaning that they get evaluated in the order they are defined before field validator gets a chance to be evaluated again according to their order defined.

Short cuircuiting and validator flavours

A FieldValidator that gets short-circuited will only prevent other FieldValidators for the same field from being evaluated. Note that this "same field" behavior applies regardless of whether the or syntax was used to declare the validation rule. By way of example, given this -validation.xml file:

```
<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>
```

both validators will be run, even if the "required" validator short-circuits. "required" validators are FieldValidator's and will not short-circuit the plain ExpressionValidator because FieldValidators only short-circuit other checks on that same field. Since the plain Validator is not field specific, it is not short-circuited.

How Validators of an Action are Found

As mentioned above, the framework will also search up the inheritance tree of the action to find default validations for interfaces and parent classes of the Action. If you are using the short-circuit attribute and relying on default validators higher up in the inheritance tree, make sure you don't accidentally short-circuit things higher in the tree that you really want!

AJAX Validation

This page last changed on Mar 04, 2007 by [maydeen](#).

Description

The following is describes how to do simple ajax validation in webwork.



This requires DWR servlet being setup, dojo and the ajax theme being used.



In the Ajax theme, dwr is used for normal validation while dojo everything else (widgets, XHR, browser JS events etc.)



In order for validation to function properly it is advised to used standard WebWork Tags.

Setup DWR

DWR could be setup by having the following dwr configuration (dwr.xml) at /WEB-INF/ directory. If it needs to be in other places, refer to <http://getahead.ltd.uk/dwr/> for more information.

```
<!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>
```

A DWRServlet need to be registered with the web application as well. The following shows a typical web.xml with DWRServlet.

```
<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>

```

Step 1

Create the jsp page. Note the <ww:head ...> tag is used to set the theme which will put in necessary dojo scripts etc. See ajax's theme head.ftl for more information.

```

<html>
<head>
    <title>Validation - Basic</title>
    <ww:head theme="ajax" debug="true"/>
</head>

<body>
<p>
This quiz (ajax) example is customized to use 2 locale, namely en_US and cn_ZH, as I don't know
how to write / read chinese, the chinese resource bundle is just like the english but prefixed
with (cn)
</p>
<ul>
    <li>
        <ww:url id="url" namespace="/validation" action="quizAjax" method="input">
            <ww:param name="request_locale" value="%{'zh_CN'}" />
        </ww:url>
        To switch to use the chinese resource bundle click <ww:a
        href="#">here</ww:a>.
    </li>
    <li>
        <ww:url id="url" namespace="/validation" action="quizAjax" method="input">
            <ww:param name="request_locale" value="%{'en_US'}" />
        </ww:url>
        To switch to use the english resource bundle click <ww:a
        href="#">here</ww:a>.
    </li>
</ul>

<p>
The following form uses the labelposition="left"
<ww:form id="f1" action="quizAjax" namespace="/validation" method="post" validate="true"
theme="ajax">
    <ww:textfield label="Name" name="name" labelposition="left" />
    <ww:textfield label="Age" name="age" labelposition="left" />
    <ww:textfield label="Favorite color" name="answer" labelposition="left" />
    <ww:submit id="b1" />
</ww:form>
</p>

<p>
The following form uses the labelposition="top"
<ww:form id="f2" action="quizAjax" namespace="/validation" method="post" validate="true"
theme="ajax">
    <ww:textfield label="Name" name="name" labelposition="top" />
    <ww:textfield label="Age" name="age" labelposition="top" />
    <ww:textfield label="Favorite color" name="answer" labelposition="top" />
    <ww:submit id="b2" />
</ww:form>
</p>

</body>
</html>

```

Step 2

Create the action class

```
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;  
    }  
}
```

Step 3

Create the 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 key="validation.name.required"/>  
        </field-validator>  
    </field>  
    <field name="age">  
        <field-validator type="int">  
            <param name="min">13</param>  
            <param name="max">19</param>  
            <message key="validation.age.invalid" />  
        </field-validator>  
    </field>  
</validators>
```

Basic Validation

This page last changed on Mar 06, 2007 by [tm_jee](#).

Description

The following are the steps to configure a basic validation using Webwork.

Step 1:

Write up a html form.

```
<html>
<head>
    <title>Validation - Basic</title>
    <ww:head/>
</head>

<body>

<p>
```

The following form uses labelposition="left"

```
<ww:form id="f0" method="post" labelposition="left">
    <ww:textfield label="Name" name="name" labelposition="left"/>
    <ww:textfield label="Age" name="age" labelposition="left"/>
    <ww:textfield label="Favorite color" name="answer" labelposition="left"/>
    <ww:submit/>
</ww:form>
</p>
```

The following form uses labelposition="top"

```
<ww:form id="f1" method="post" labelposition="top">
    <ww:textfield label="Name" name="name" labelposition="top"/>
    <ww:textfield label="Age" name="age" labelposition="top"/>
    <ww:textfield label="Favorite color" name="answer" labelposition="top"/>
    <ww:submit/>
</ww:form>
</p>
```

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

Step 2:

Write up the action class. Get methods must be supplied for validation to work.

```
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;
}
}

```

Step3:

Write up the validators to be used. The validation.xml format is either <ActionClassName>-validation.xml or <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 key="validation.name.required"/>
        </field-validator>
    </field>
    <field name="age">
        <field-validator type="int">
            <param name="min">13</param>
            <param name="max">19</param>
            <message key="validation.age.invalid" />
        </field-validator>
    </field>
</validators>

```

Client Side Validation

This page last changed on Jan 07, 2006 by [plightbo](#).

Description

WebWork adds support for client-side validation on top of XWork's standard validation framework. It can be enabled on a per-form basis by specifying **validate="true"** in the [form](#) tag:

```
<ww:form name="test" action="javascriptValidation" validate="true">
  ...
</ww:form>
```



A name for the form can be supplied, else the action name will be used as the form name.

A correct *action* and *namespace* attributes must be provided to the [<ww:form>](#) tag. For example, the Action named "submitProfile" is used in the "/user" namespace, the following must be used.

```
<ww:form namespace="/user" action="submitProfile" validate="true">
  ...
</ww:form>
```

While the following will "work" in the sense that the form will function correctly, client-side validation will not. That is because WebWork must know the exact namespace and action (rather than a URL) to properly support validation.

```
<ww:form action="/user/submitProfile.action" validate="true">
  ...
</ww:form>
```

All the standard [validation configuration](#) steps still apply. Client-side validation uses the same validation rules as server-side validation. If server-side validation doesn't work, then client-side validation won't work either.



i Note that the *required* attribute on many WebWork [Tags](#) has nothing to do with client-side validation. It is just used by certain theme (like xhtml) to put a '*' on the field marked as required.

Client Side Validation Types

There are two styles of client side validation:

- [Pure JavaScript Client Side Validation](#) - used by the [xhtml theme](#) and [css_xhtml theme](#)
- [AJAX Client Side Validation](#) - used by the [ajax theme](#)

AJAX Client Side Validation

This page last changed on Jan 14, 2007 by [tm_jee](#).

AJAX-based client side validation improves upon [Pure JavaScript Client Side Validation](#) by using a combination of JavaScript, DOM manipulation, and remote server communication via [DWR](#). Unlike the pure client side implementation, AJAX-based validation communicates with the server. This means all your validation rules that worked when submitting a form will still work within the browser.



This validation mode only works with the [ajax theme](#)

The validation occurs on each **onblur** event for each form element. As each user types in some values and moves to the next form element, the value (and all other values previously entered) will be sent to the server for validation. The entire validation stack is run, including visitor validators and your action's validate() method.

If there is an error, like the pure implementation, the HTML and DOM will be updated immediately.

For an example of this, see [AJAX Validation](#).



A description of the internal workings of WebWork Ajax Validation could be found [here](#)

Workings Of WebWork Ajax Validation

This page last changed on Jan 23, 2007 by [tm_jee](#).

WebWork uses DWR to implement its ajax validation feature.

To have DWR configured, the following snippet needs to be in web.xml

An error occurred:

<http://svn.opensymphony.com/svn/webwork/webapps/showcase/src/webapp/WEB-INF/web.xml>. The system administrator has been notified.

and dwr.xml under WEB-INF should contains snippet like

An error occurred:

<http://svn.opensymphony.com/svn/webwork/webapps/showcase/src/webapp/WEB-INF/dwr.xml>. The system administrator has been notified.

With the above configuration, WebWork would be able to call

```
validator.doPost(
    function(validationAwareSupportReturned) {
        ...
    },
    action, namespace, params);
```

to invoke WebWork's DWRValidator's doPost(action, namespace, params) and processed the ValidationAwareSupport returned.

When there's a change on webwork ui components, WebWork will make such an ajax call, (as shown in /template/ajax/validation.js)

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/ajax/validation.js>. The system administrator has been notified.

A new instance of ValidationClient js function is created for each form and the internal of contacting dwr is done there as well. Bellow is snippet of validationClient.js located in /static/validationClient.js

An error occurred:

<http://svn.opensymphony.com/svn/webwork/src/java/com/opensymphony/webwork/static/validationClient.js>. The system administrator has been notified.

In ValidationClient.js function there's an empty implementation of onErrors, this is where codes of update the ui when there's an error etc. WebWork's ajax's theme actually overrides this with custom hooks as shown in /template/ajax/validation.js. Therefore a particular theme would need to implement the following functions to suite their own needs.

```
/**
 * Clear error message from the <code>form</code>
 */
function clearErrorMessages(form) {
    ...
}

/*
 * Clear the error label from <code>form</code>, eg. change the style so they don't appear
 as red in color
```

```
/*
function clearErrorLabels(form) {
}

/*
 * Add error <code>errorOfFieldElement</code> to <code>fieldElement</code>.
 */
function addError(fieldElement, errorOfFieldElement) {
}
```

An example of xhtml's implementation of the above functions is as follows:-

An error occurred: <http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/validation.js>.
The system administrator has been notified.

Pure JavaScript Client Side Validation

This page last changed on Jan 10, 2007 by [phil](#).

Pure JavaScript client side validation is the simplest but least feature-rich type of [Client Side Validation](#). This type of validation uses 100% client-side JavaScript code to try to validate the values entered by the user. Because the validation logic is actually repeated in the JavaScript code, it is important to understand that some values will be considered acceptable by the JavaScript code but will be marked as unacceptable by the server-side [Validation](#).

An error occurred:

<http://svn.opensymphony.com/svn/webwork/src/java/template/xhtml/form-close-validate.ftl>. The system administrator has been notified.

Error reporting

Because client side validation does not talk to the server, the theme ([xhtml theme](#) or [css xhtml theme](#)) is responsible for properly manipulating the HTML DOM to display the error message inline. The JavaScript that is responsible for doing this logic is **validation.js** and can be found in each theme.



Errors are reported using the default validation message, not the internationalized version that the server-side might be aware of. This is a known issue. You may want to try the [AJAX Client Side Validation](#) for messages that are fully internationalized.

Additional Validator Support

If you wish to add additional validator support beyond those listed, you may override the [xhtml theme](#) template **form-close-validate.ftl**. This file contains the JavaScript that tries to validate each user-entered value from within the browser. The [css xhtml theme](#) extends the [xhtml theme](#) and therefore doesn't have its own form-close-validate.ftl template.

Client Validation

This page last changed on Mar 04, 2007 by [maydeen](#).

Description

The following is the step to create a Client-Side validation using webwork. Note the validate attribute is set to true. Note also that not all themes support this feature (client-side validation)

Step 1

Create the jsp page. Note the <ww:head> tag being used, it will setup the css in this case (xhtml theme)

```
<html>
<head>
    <title>Validation - Basic</title>
    <ww:head />
</head>

<body>

<p>
The following form uses labelposition="left"
<ww:form id="f1" name="quizClient" namespace="/validation" method="post" validate="true">
    <ww:textfield label="Name" name="name" labelposition="left" />
    <ww:textfield label="Age" name="age" labelposition="left" />
    <ww:textfield label="Favorite color" name="answer" labelposition="left" />
    <ww:submit/>
</ww:form>
</p>
<br/>

<p>
The following form uses labelposition="top"
<ww:form id="f2" name="quizClient" namespace="/validation" method="post" validate="true">
    <ww:textfield label="Name" name="name" labelposition="top" />
    <ww:textfield label="Age" name="age" labelposition="top" />
    <ww:textfield label="Favorite color" name="answer" labelposition="top" />
    <ww:submit/>
</ww:form>
</p>

</body>
</html>
```

Step 2

Create the action class

```
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;
    }
}

```

Step 3

Create the validation.xml to configure the validators to be used.

```

<!--
    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 key="validation.name.required"/>
        </field-validator>
    </field>
    <field name="age">
        <field-validator type="int">
            <param name="min">13</param>
            <param name="max">19</param>
            <message key="validation.age.invalid" />
        </field-validator>
    </field>
</validators>

```

collection validator

This page last changed on Jan 10, 2007 by [phil](#).



This is an experimental feature. Feedback welcome !

Description

Validate a property available in the object of a collection.

Parameters

- property - the full property name that this validator should validate. eg. if "persons" is a collection of Persons object with a property called name the property would be "persons.name"
- validatorRef - validator name of an existing validator (could not be "collection" validator) eg. required, requiredstring, int etc.
- validatorParams - the parameters to be passed into the validator referenced by the "validatorParams" attribute above.

Examples

```
public class MyAction extends ActionSupport {
    private List persons = new ArrayList();
    ...
    public List getPersons() { return this.persons; }
    public void setPersons(List persons) { this.persons = persons; }
}

public class Person {
    private String name;
    private Integer age;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }

    public Integer getAge() { return age; }
    public void setAge(Integer age) { this.age = age; }
}

<validators>
<field name="persons">
    <field-validator type="collection">
        <param name="property">persons.name</param>
        <param name="validatorRef">requiredstring</param>
        <param name="validatorParams['defaultMessage']">Must be String</param>
        <param name="validatorParams['trim']">true</param>
        <message></message>
    </field-validator>
    <field-validator type="collection">
        <param name="property">persons.age</param>
        <param name="validatorRef"></param>
        <param name="validatorParams['defaultMessage']">Must be filled in</param>
```

```
<message></message>
</field-validator>
<field-validator type="collection">
    <param name="property">persons.age</param>
    <param name="validatorRef">int</param>
    <param name="validatorParams['defaultMessage']">Needs to be an integer</param>
    <message></message>
</field-validator>
</field>
</validators>
```

conversion validator

This page last changed on Apr 05, 2006 by [tm_jee](#).

Description

Field Validator that checks if a conversion error occurred for this field.

Parameters

- `fieldName` - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required

Examples

```
<pre>
    <!-- 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>
</pre>
```

Repopulating Field upon conversion Error

The capability of auto-repopulating the stack with a fake parameter map when a conversion error has occurred can be done with 'repopulateField' property set to "true".

This is typically usefull when one wants to repopulate the field with the original value when a conversion error occurred. Eg. with a textfield that only allows an Integer (the action class have an Integer field declared), upon conversion error, the incorrectly entered integer (maybe a text 'one') will not appear when dispatched back. With 'repopulateField' porperty set to true, it will, meaning the textfield will have 'one' as its value upon conversion error.

```
<!-- 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 Dec 11, 2005 by [tm_jee](#).

Description

Field Validator that checks if the date supplied is within a specific range.

NOTE: If no date converter is specified, XWorkBasicConverter will kick in to do the date conversion, which by default using the `Date.SHORT` format using the locale specified in `webwork.properties` else falling back to the system default locale.

Parameters

- `fieldName` - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required
- `min` - the min date range. If not specified will not be checked.
- `max` - the max date range. If not specified will not be checked.

Examples

```
<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-validator>
    </field>
</field>
</validators>
```

email validator

This page last changed on Dec 11, 2005 by [tm_jee](#).

Description

EmailValidator checks that a given String field, if not empty, is a valid email address. The regular expression used to validate that the string is an email address is:

```
\b([_A-Za-z0-9-]+(\.[_A-Za-z0-9-]*@[A-Za-z0-9-]+(\.\com|\.\net|\.\org|\.\info|\.\edu|\.\mil))
```

Parameters

- **fieldName** - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required

Examples

```
<!-- 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 Dec 11, 2005 by [tm_jee](#).

Description

A Non-Field Level validator that validates based on regular expression supplied.

Parameters

- expression - the Ognl expression to be evaluated against the stack (Must evaluate to a Boolean)

Examples

```
<validators>
    <validator type="expression">
        <param name="expression"> .... </param>
        <message>Failed to meet Ognl Expression .... </message>
    </validator>
</validators>
```

fieldexpression validator

This page last changed on Dec 11, 2005 by [tm_jee](#).

Description

Validates a field using an OGNL expression.

Parameters

- **fieldName** - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required
- **expression** - The Ognl expression (must evaluate to a boolean) which is to be evaluated the stack

Examples

```
<!-- 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>
</validators>
```

int validator

This page last changed on Dec 11, 2005 by [tm_jee](#).

Description

Field Validator that checks if the integer specified is within a certain range.

Parameters

- fieldName - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required
- min - the minimum value (if none is specified, it will not be checked)
- max - the maximum value (if none is specified, it will not be checked)

Examples

```
<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 Dec 11, 2005 by [tm_jee](#).

Description

Validates a string field using a regular expression.

Parameters

- **fieldName** - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required
- **expression** - The RegExp expression REQUIRED
- **caseSensitive** - Boolean (Optional). Sets whether the expression should be matched against in a case-sensitive way. Default is `true`.

Examples

```
<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 Dec 11, 2005 by [tm_jee](#).

Description

RequiredFieldValidator checks if the specified field is not null.

Parameters

- fieldName - field name if plain-validator syntax is used, not needed if field-validator syntax is used

Examples

```
<validators>

    <!-- Plain Validator Syntax -->
    <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 Dec 11, 2005 by [tm_jee](#).

Description

RequiredStringValidator checks that a String field is non-null and has a length > 0. (i.e. it isn't ""). The "trim" parameter determines whether it will {@link String#trim()} the String before performing the length check. If unspecified, the String will be trimmed.

Parameters

- fieldName - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required
- trim - trim the field name value before validating (default is true)

Examples

```
<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 Dec 11, 2005 by [tm_jee](#).

Description

StringLengthFieldValidator checks that a String field is of a certain length. If the "minLength" parameter is specified, it will make sure that the String has at least that many characters. If the "maxLength" parameter is specified, it will make sure that the String has at most that many characters. The "trim" parameter determines whether it will {@link String#trim() trim} the String before performing the length check. If unspecified, the String will be trimmed.

Parameters

- **fieldName** - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required
- **maxLength** - The max length of the field value. Default ignore.
- **minLength** - The min length of the field value. Default ignore.
- **trim** - Trim the field value before evaluating its min/max length. Default true

Examples

```
<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>
    </validators>
```

url validator

This page last changed on Jan 08, 2006 by [tm_jee](#).

Description

URLValidator checks that a given field is a String and a valid URL

Parameters

- fieldName - The field name this validator is validating. Required if using Plain-Validator Syntax otherwise not required

Examples

```
<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 Mar 04, 2007 by [maydeen](#).

Description

The followings show a simple example using Webwork's Field Validators

Step 1

Create the jsp page

```
<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>
```

Step 2

Create the action class

```
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;
}
}

```

Step 3

Create the 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 Mar 04, 2007 by [maydeen](#).

Description

The followings show a simple example using Webwork's Non Field Validators

Step 1

Create the jsp page

```
<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>
```

Step 2

Create the action class

```
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;
    }
}
```

Step 3

Create the 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 Mar 04, 2007 by [maydeen](#).

Description

The followings show a simple example using Webwork's Field Validators

Step 1

Create the jsp page

```
<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>
```

Step 2

Create the action class

```
public class VisitorValidatorsExampleAction extends AbstractValidationActionSupport {

    private User user;

    public User getUser() {
        return user;
    }

    public void setUser(User user) {
        this.user = user;
    }
}
```

Step 3

Create the 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 Dec 11, 2005 by [tm_jee](#).

Description

The VisitorFieldValidator allows you to forward validation to object properties of your action using the object's own validation files. This allows you to use the ModelDriven development pattern and manage your validations for your models in one place, where they belong, next to your model classes. The VisitorFieldValidator can handle either simple Object properties, Collections of Objects, or Arrays.

Parameters

- fieldName - field name if plain-validator syntax is used, not needed if field-validator syntax is used
- context - the context of which validation should take place. If it is not used, will default to the name of the action. Optional
- appendPrefix - (true/false) determine if prefix is to be added to field. Optional

Examples

```
<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>
```

In the example above, if the action's `getUser()` method return `User` object, WebWork will look for `User-myContext-validation.xml` for the validators. Since `appendPrefix` is true, every field name will be prefixed with 'user' such that if the actual field name for 'name' is 'user.name'

Velocity

This page last changed on Mar 21, 2006 by [phil](#).

Velocity is a template Java template language.

For more information on Velocity itself, please visit the [Velocity website](#).



Velocity is very similar to [FreeMarker](#), as both are template languages that can be used outside of a Servlet container. The WebWork team recommends FreeMarker over Velocity simply because FreeMarker has better error reporting, support for JSP tags, and slightly better features. However, both are good alternatives to JSP.

Getting Started

Getting started with Velocity is as simple as ensuring all the [dependencies](#) are included in your project's classpath. Other than that, [webwork-default.xml](#) already configures the [Velocity Result](#) needed to map your actions to your templates. You may now try out the following **xwork.xml** configuration:

```
<action name="test" class="com.acme.TestAction">
    <result name="success" type="velocity">test-success.vm</result>
</action>
```

Then in **test-success.vm**:

```
<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.

Template Loading

WebWork looks for Velocity templates in two locations (in this order):

1. Web application
2. Class path

This ordering makes it ideal for providing templates inside a fully built jar, but allowing for overrides of those templates to be defined in your web application. In fact, this is how you can override the default UI

tags and [Form Tags](#) included with WebWork.

Variable Resolution

In Velocity, variables are looked up in several different places, in this order:

1. The value stack
2. The action context
3. Built-in variables

Note that the action context is looked up after the value stack. This means that you can reference the variable without the typical preceding has marker (#) like you would have to when using the JSP `ww:property` tag. This is a nice convenience, though be careful because there is a small chance it could trip you up.

```
#wwurl "id=url "value=http://www.yahoo.com"
Click <a href="${url}">here</a>!
```

The built-in variables that WebWork-Velocity integration provides are:

Name	Description
stack	The value stack itself, useful for calls like <code> \${stack.findString('ognl expr')}</code>
action	The action most recently executed
response	The <code>HttpServletResponse</code>
res	Same as response
request	The <code>HttpServletRequest</code>
req	Same as request
session	The <code>HttpSession</code>
application	The <code>ServletContext</code>
base	The request's context path

Tag Support

See the [Velocity Tags](#) documentation for information on how to use the generic [Tags](#) provided by WebWork.

Tips and Tricks

There are some advanced features that may be useful when building WebWork applications with Velocity.

Extending

Sometimes you may wish to extend the Velocity support provided with WebWork. The most common reason for doing this is that you wish to include your own [Tags](#), such as those that you have extended from the built in WebWork [Tags](#).

To do so, write a new class that extends

com.opensymphony.webwork.views.velocity.VelocityManager and overrides it as needed. Then add the following to [webwork.properties](#):

```
webwork.velocity.manager.classname = com.yourcompany.YourVelocityManager
```

Configuring

You can configure Velocity by placing configuration items in [velocity.properties](#).

velocity.properties

This page last changed on Dec 11, 2005 by [plightbo](#).

This file if provided (/WEB-INF/classes) will be loaded by Velocity. It can be used to load custom macros:

```
# Velocity Macro libraries.  
velocimacro.library = webwork.vm, tigris-macros.vm, myapp.vm
```

See the [Velocity](#) documentation for more information.

XWork Configuration

This page last changed on Mar 21, 2006 by [phil](#).

Overview

All action configuration is done from within [xwork.xml](#) (see [Configuration Files](#) for more info). In this section we discuss the various elements that make up the action configuration, such as actions, interceptors, results, and package.

1. [Package Configuration](#)
2. [Namespace Configuration](#)
3. [Include configuration](#)
4. [Action configuration](#)
5. [Result Configuration](#)
6. [Interceptor Configuration](#)
7. [Views](#)

Action configuration

This page last changed on Mar 21, 2006 by [tm_jee](#).

Actions are the basic "unit-of-work" in WebWork, they define, well, actions. An action will usually be a request, (and usually a button click, or form submit). The main action element (tag is too synonymous with JSP) has two parts, the friendly name (referenced in the URL, i.e. saveForm.action) and the corresponding "handler" class.

```
<action name="formTest" class="com.opensymphony.webwork.example.FormAction"
method="processForm">
```

The optional "**method**" parameter tells WebWork which method to call based upon this action. If you leave the method parameter blank, WebWork will call the method **execute()** by default. If there is no execute() method and no method specified in the xml file, WebWork will throw an exception.

Also, you can tell WebWork to invoke "**doSomething**" method in your action by using the pattern "**actionName!something**" in your form. For example, "**formTest!save.action**" will invoke the method "**save**" in FormAction class. The method must be public, take no arguments and also returns a String which indicate the name of the result to be executed:

```
public String save() throws Exception
{
    ...
    return SUCCESS;
}
```

All the configuration for "**actionName**" will be used for "actionName!something" (interceptors, result types, etc...)

Action Support

Action class attribute could be left out such as following

```
<action name="myAction">
    ...
</action>
```

In this case, the class will default to com.opensymphony.xwork.ActionSupport which have an execute() method that returns SUCCESS by default.

Default Action Reference

Since Webwork 2.2.1 you are also able to specify a default action to be executed when no other action is matched in the xwork.xml. This feature is provided mainly to eliminate the need to create an action class

and/or element for very simple or similar results. The default action name can be set inside the package element like below:

```
<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>
```



Caution

This feature should be configured such that there is only one default action per namespace. Therefore if you have multiple packages declaring a default action with the same namespace, it is not guaranteed which action will be the default.



Note

Note that the name attribute is left out for the first result, as WebWork will default to "success" if it is left out.

In this case any request to action not defined in this package will automatically trigger action with alias "simpleViewResultAction" to be executed.

Most of the content here provided by Matt Dowell <matt.dowell@notiva.com>

Include configuration

This page last changed on Jan 01, 2006 by [tm_jee](#).

Description

To make it easy to manage large scale development (lots of actions + configuration), WebWork allows you to include other configuration files from xwork.xml :

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

The included files must be the same format as xwork.xml (with the doctype and everything) and be placed on classpath (usually in /WEB-INF/classes or jar files in /WEB-INF/lib).

Most of the content here provided by Matt Dowell <matt.dowell@notiva.com>

Interceptor Configuration

This page last changed on Jan 01, 2006 by [tm_jee](#).

Description

Interceptors allow you to define code to be executed before and/or after the execution of an action. Interceptors can be a powerful tool when writing web applications. Some of the most common implementations of an Interceptor might be:

- Security Checking (ensuring the user is logged in)
 - Trace Logging (logging every action)
 - Bottleneck Checking (start a timer before and after every action, to check bottlenecks in your application)
- You can also chain Interceptors together to create an interceptor **stack**. If you wanted to do a login check, security check, and logging all before an Action call, this could easily be done with an interceptor package.

Interceptors must first be defined (to give name them) and can be chained together as a stack:

```
<interceptors>
    <interceptor name="security" class="com.mycompany.security.SecurityInterceptor"/>
    <interceptor-stack name="defaultComponentStack">
        <interceptor-ref name="component" />
        <interceptor-ref name="defaultStack" />
    </interceptor-stack>
</interceptors>
```

To use them in your actions:

```
<action name="VelocityCounter" class="com.opensymphony.webwork.example.counter.SimpleCounter">
    <result name="success">...</result>
    <interceptor-ref name="defaultComponentStack" />
</action>
```

NOTE: Reference name can be either the name of the interceptor or the name of a stack

For more details, see [Interceptors](#) reference.

Most of the content here provided by Matt Dowell <matt.dowell@notiva.com>

Namespace Configuration

This page last changed on Jan 01, 2006 by [tm_jee](#).

Namespaces

The namespace attribute allows you to segregate action configurations into namespaces, so that you may use the same action alias in more than one namespace with different classes, parameters, etc. This is in contrast to Webwork 1.x, where all action names and aliases were global and could not be re-used in an application.

Default Namespace

The default namespace, which is "" (an empty string) is used as a "catch-all" namespace, so if an action configuration is not found in a specified namespace, the default namespace will also be searched. This allows you to have global action configurations outside of the "extends" hierarchy, as well as to allow the previous Webwork 1.x behavior by not specifying namespaces. It is also intended that the namespace functionality can be used for security, for instance by having the path before the action name be used as the namespace by the Webwork 2.0 ServletDispatcher, thus allowing the use of J2EE declarative security on paths to be easily implemented and maintained.

Root Namesapce

Root namespace, which is "/" is also allowed in WebWork. It will be the namespace when a request directly under the context path is received. As with other namespace, it will fall back to the default namespace if no such action alias is found in it.

Namespace example

```
<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.simpleAction">
        <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>
```

Explanation

If a request for /barspace/bar.action is made, '/barspace' namespace is searched and if it is found the bar action is executed, else it will fall back to the default namespace. In this example bar alias do exists in the '/barspace' namespace, so it will get executed and if success, the request will be forwarded to bar2.jsp.

Note: If a request is made to /barspace/foo.action, the action foo will be searched for in a namespace of /barspace. If the action is not found, the action will then be searched for in the default namespace. Unless specified, the default namespace will be "". In our example above, their is no action foo in the namespace /barspace, therefore the default will be searched and /foo.action will be executed.

If a request for /moo.action is made, the root namespace ('/') is searched for 'moo' action alias, if it is not found it will fall back to trying to find it in the default namespace. In this example, moo action alias does exists and hence will be executed. Upon sucess, the request will get forwarded to bar2.jsp.

Note: If a request is made to '/foo.action', '/' namespace will be searched and if it is found it will be executed, else an attempt to try to find it in the default namespace will occurred. In this example foo action alias does not exist in the '/' namespace, hence it will failed back to the default namespace and execute the foo alias there.

Note: Namespace is only one level. For example if the url '/barspace/myspace/bar.action' is requested, Webwork will try to search for namespace '/barspace/myspace', which does not exist in this case, and will fall back to the default namespace "" and tried the search for action with 'bar' alias. As a result the bar action in the default will be used.

Package Configuration

This page last changed on Jan 06, 2006 by [tm_jee](#).

Overview

Packages are a way to group Actions, Results, Result Types, Interceptors and Stacks into a logical unit that shares a common configuration. Packages are similar to objects in that they can be extended and have individual parts overridden by "sub" packages.

Packages

The package element has one required attribute, "name", which acts as the key for later reference to this package. The "extends" attribute is optional and allows one package to inherit the configuration of one or more previous packages including all interceptor, interceptor-stack, and action configurations. Note that the configuration file is processed sequentially down the document, so the package referenced by an "extends" should be defined above the package which extends it. The "abstract" optional attribute allows you to make a package abstract, which will allow you to extend from it without the action configurations defined in the abstract package actually being available at runtime.

Attribute	Required	Description
name	yes	key to for other packages to reference
extends	no	inherits package behavior of the package it extends
namespace	no	see Namespace Configuration
abstract	no	declares package to be abstract (no action configurations required in package)

Sample usage of packages in xwork.xml

An error occurred:

<http://svn.opensymphony.com/svn/webwork/webapps/showcase/src/webapp/WEB-INF/classes/xwork.xml>.
The system administrator has been notified.

Result Configuration

This page last changed on Jan 01, 2006 by [tm_jee](#).

Description

Results are string constants that Actions return to indicate the status of an Action execution. A standard set of Results are defined by default: error, input, login, none and success. Developers are, of course, free to create their own Results to indicate more application specific cases. Results are mapped to defined [Result Types](#) using a name-value pair structure.

- [Global results](#)
- [Default results](#)

Result tags

Result tags tell WebWork what to do next after the action has been called. There are a standard set of result codes built-in to WebWork, (in the Action interface) they include:

```
String SUCCESS = "success";
String NONE = "none";
String ERROR = "error";
String INPUT = "input";
String LOGIN = "login";
```

You can extend these as you see fit. Most of the time you will have either **SUCCESS** or **ERROR**, with **SUCCESS** moving on to the next page in your application;

```
<result name="success" type="dispatcher">
    <param name="location">/thank_you.jsp</param>
</result>
```

...and **ERROR** moving on to an error page, or the preceding page;

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

Results are specified in a xwork xml config file(xwork.xml) nested inside `<action>`. If the location param is the only param being specified in the result tag, you can simplify it as follows:

```
<action name="bar" class="myPackage.barAction">
    <result name="success" type="dispatcher">
        <param name="location">foo.jsp</param>
    </result>
</action>
```

or simplified

```
<action name="bar" class="myPackage.barAction">
    <result name="success" type="dispatcher">foo.jsp</result>
</action>
```

or even simplified further

```
<action name="bar" class="myPackage.barAction">
    <result>foo.jsp</result>
</action>
```



Default Action Class

If the class attribute is not specified in the action tag, it will default to WebWork's ActionSupport.



Default Location Parameter

If no param tag eg. `<param name="location"> ,,</param>` is given as child of the `<result ...>` tag, WebWork will assume the text enclosed within the `<result> </result>` tags to be the location.



Default Result Type

If no type attribute is specified in the `<result ...>` tag, WebWork assume the type to be dispatcher. (Analogous to Servlet's Specs. SerlvetDispatcher's forward).

Default results

This page last changed on Jan 01, 2006 by [tm_jee](#).

Description

Webwork has the ability to define a default result type for your actions. Thus, you don't have to specify the result-type for results using the default. If a package extends another package and you don't specify a new default result type for the child package, then the parent package default type will be used when the type attribute is not specified in the result tag.

```
<!-- 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 Jan 01, 2006 by [tm_jee](#).

Description

Global results allows you to define result mappings which will be used as defaults for all action configurations and will be automatically inherited by all action configurations in this package and all packages which extend this package. In other words, if you have the same result specified within multiple actions, then you can define it as a global result.

Example

```
<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>
```

Same thing

```
<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 Jan 01, 2006 by [tm_jee](#).

Description

WebWork supports JSP and Velocity for your application presentation layer. For this example we will use a JSP file. Webwork comes packaged with a tag library (taglibs). You can use these taglibs as components in your JSP file. Here is an section of our form.jsp page:

```
<%@ 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>
```

The process of events will go as follows:

1. WebWork will take notice since the URI ends in **.action** (defined in **web.xml** files)
2. WebWork will look up the action **formTest** in its action hierarchy and call any Interceptors that might have been defined.
3. WebWork will translate **formTest** and decide to call the method **processForm** in the class **com.opensymphony.webwork.example.FormAction** as defined in **xwork.xml** file.
4. The method will process successfully and give WebWork the **SUCCESS** return parameter.
5. WebWork will translate the **SUCCESS** return parameter into the location **formSuccess.jsp** (as defined in **xwork.xml**) and redirect accordingly.

Most of the content here provided by Matt Dowell <matt.dowell@notiva.com>

Related Tools

This page last changed on Jun 07, 2006 by [aramati](#).

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](#)
2. [Debugging inside WebWork - Debuggability of your Application](#)

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 Dec 29, 2005 by [niko2416](#).

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>.

Debugging inside WebWork - Debuggability of your Application

This page last changed on Jun 07, 2006 by [aramati](#).

Debuggability is a feature that is very useful and also necessary to have inside of WebWork. The config-browser lets you browse through your actions and see your properties and several other information.

But what about seeing the values of your properties or testing your OGNL-expressions against the value stack?

If you like to do that, use the Debugging Interceptor.
It is already implemented in Struts, but I included it into Webwork 2.2.2.

Installing Debuggability

Download the Attachment "Debuggability for WebWork 2.2.2-all files.zip" before following the installing instructions.

1. Add the folder hierarchy "**/interceptors/debugging**" to your application.
2. Compile the files "**DebuggingInterceptor.java**" and "**PrettyPrintWriter.java**", after you have customised the package-path
inside these two java files to refer to your application' package "/interceptors/debugging".
Add these files to your package "interceptors.debugging".
3. Add the "**console.ftl**" (Freemarker-template) to a folder called "templates/freemarker" to the root of your webapplication.
If you like to put it someplace else you have to change the FreemarkerResult-location-variable in DebuggingInterceptor.java (in my application it is done by result.setLocation("/templates/freemarker/console.ftl").)
Here you should save the place of the console.ftl.
4. Tell the console.ftl the **baseUrl** to search for its files by defining

```
var baseUrl = "/yourApplication-Root/templates";
```

5. Save the following files in **/yourApplication-Root/templates**:

- *webconsole.css* - the stylesheet for the console
- *webconsole.jsp* - the jsp to show the console and to make input and outputs available
- *webconsole.js* - the javascript-file to make key-events available and to print the results of your inputs

6. Your **pom.xml** should make freemarker available:

```

<dependency>
  <groupId>freemarker</groupId>
  <artifactId>freemarker</artifactId>
  <version>2.3.4</version>
</dependency>

```

7. Your **webwork.properties** needs to allow debugging by adding the following line:

```
webwork.devMode=true
```

8. To use the interceptor, add it to your **xwork.xml** with a reference:

```
<interceptor-ref name="debugging" />
```

9. **Declare the interceptor** to your application by

```

<interceptors>
  <interceptor name="debugging"
  class="com.agimatec.ostium.portlets.observator.interceptors.debugging.DebuggingInterceptor"/>
</interceptors>

```

(e.g. inside the package-Tag).

For the class of the interceptor you have - of course - customise the path of the DebuggingInterceptor-class!

If you have any problems or further questions, contact me, Tamara Cattivelli, at aramati@web.de.

Using Debugger

You have several possibilities to use the Debugger, you "installed" right now.

If you like to debug your application/action, just call your action (if you like through the config-browser) and add the debug-parameter to your action query string, by appending "?" and then the value.

value	description
debug=xml	to get an xml dump of your Action on the page
debug=console	to test OGNL expressions against the value stack

example:

```
http://localhost:8080/mypath/tomy/applicationsaction/myAction.action?debug=console
```


EclipseWork

This page last changed on Oct 30, 2005 by [phil](#).

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](#).

IDEA Plugin

This page last changed on Mar 23, 2006 by [phil](#).



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:

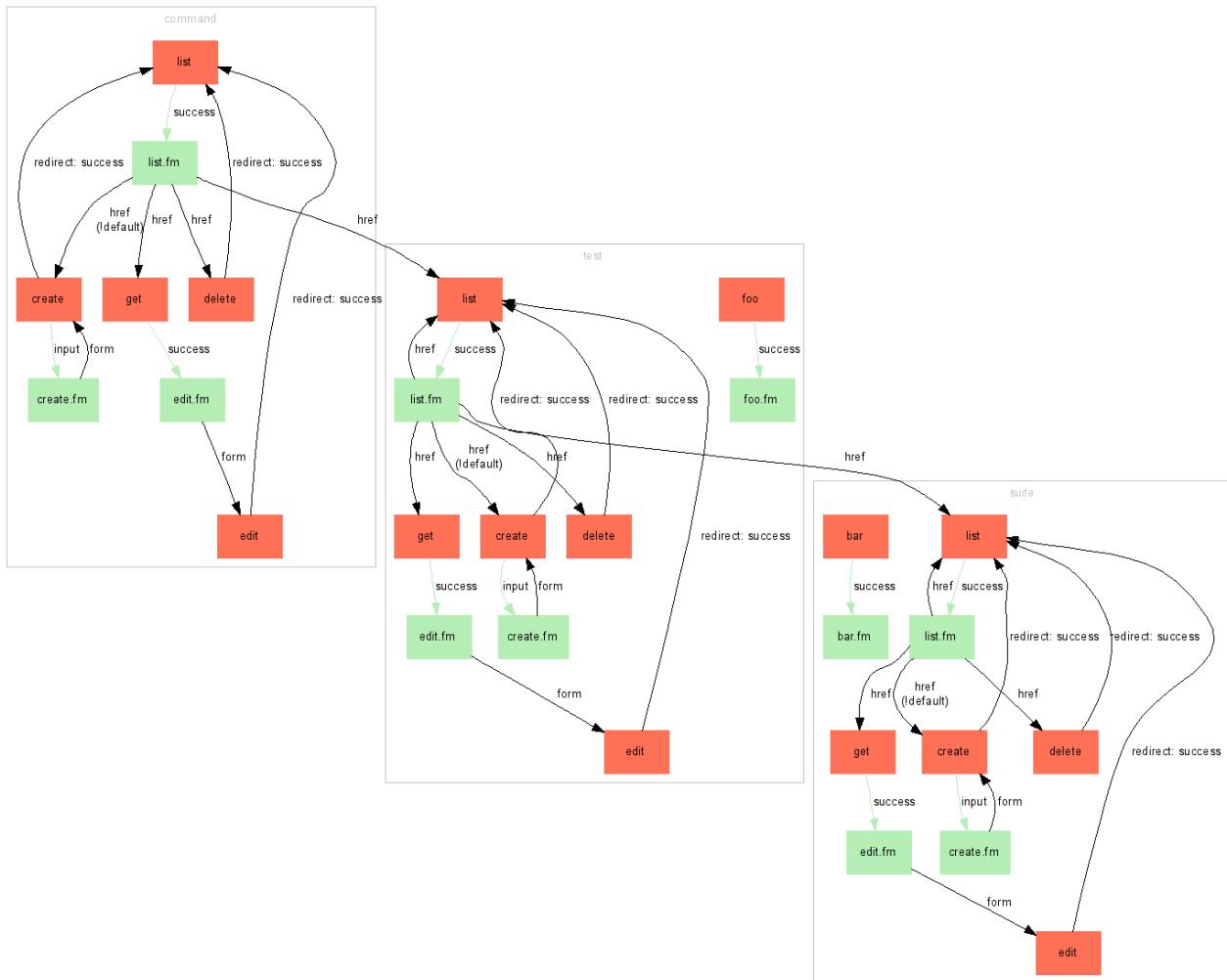
- **Configuration browser** - you can browse namespaces and see both actions and views as well as their relationships. The browser lets you avoid hunting through xwork.xml and all the included files and instead presents a logical view of all your actions and their associated views. The browser also allows 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 "dокумент" 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.

SiteGraph

This page last changed on Dec 11, 2005 by [plightbo](#).

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](#)):



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:

Error formatting macro: snippet: java.lang.IllegalArgumentException: Invalid url: must begin with a configured prefix.

- You must supply the correct classpath when invoking the SiteGraph tool. Specifically, the XWork, WebWork, and their dependencies must be included in the classpath. Furthermore, **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();
```

ValueStack Explorer

This page last changed on Dec 05, 2006 by [phil](#).

The ValueStack Explorer

Introduction

One of the most powerful parts of WebWork 2 and Struts 2 is the OGNL ValueStack, a stack which can be queried with OGNL expressions. Unfortunately enough, this very same part is the cause of much confusion amongst new users of the framework. There is a lot of information exposed to the view layer, but often it requires exploring the stack by a number of iterators. Testing expressions takes time, and sometimes a tiny typo can cause much grief.

What is it ?

The ValueStack Explorer is a combination of 2 tags (breakpoint and snapshot), a switch, and a monitoring client. The client allows you to send expressions to the breakpoint tag, which will block the current execution and send the response to the client. As such, it effectively operates as a breakpoint like any standard debugger.

How does it work ?

You start by starting the Switch - which is nothing more than a very simple message relay (=chat server). It accepts connections on a socket (by default it runs on port 4281). The client, created in Adobe Flash (should be ported to Java Swing - one day ..) connects to the Switch, can send OGNL statements and parses incoming (XML wrapped) responses. The breakpoint tag, when active, will open a socket and connect to the Switch as well, and execute any command against the OGNL Value Stack.

Installation

Download the first binary release here:

<http://wiki.opensymphony.com/download/attachments/8204/explorer-0.1.zip> (source is coming, need to check Flash sources).

Unzip the explorer-x.x.zip file.

Place the explorer.jar file in your webapplication's WEB-INF/lib directory. Doubleclick it (or execute java -jar explorer.jar in a terminal) to start the switch.

Either run the client/explorer.swf file directly or open the client/explorer.html page. You should see the Flash client (make sure you have Flash 7+ plugin installed). It will connect by default to localhost on port 4281. If you want it to connect to another host, you'll have to provide the parameters yourself in the explorer.html file.

Open the explorer.html file in your favorite text editor, and change **both** explorer.swf?host=127.0.0.1&port=4281 (one for firefox/Opera, one for IE) references to whatever you

want:

```
..  
<param name="movie" value="explorer.swf?host=127.0.0.1&port=4281" />  
..  
<embed src="explorer.swf?host=127.0.0.1&port=4281" quality="high" bgcolor="#ffffff"  
width="1000" height="600" name="explorer" align="middle" allowScriptAccess="sameDomain"  
type="application/x-shockwave-flash" pluginspage="http://www.macromedia.com/go/getflashplayer"  
/>  
..
```

Tutorial

Create a simple action called TestAction.

```
..  
public class TestAction extends ActionSupport {  
  
    private List list;  
  
    public String execute() throws Exception {  
  
        list = new ArrayList();  
        list.add("Rene");  
        list.add("Peter");  
        list.add("Toby");  
        list.add("Rainier");  
        list.add("Patrick");  
  
        return super.execute();  
    }  
  
    public List getList() {  
        return list;  
    }  
}
```

Create a test page test.jsp which will display the list of names, and place it under the /WEB-INF/jsp/ directory.

```
<%@ taglib uri="/webwork" prefix="ww" %>  
<%@ taglib uri="/vs" prefix="vs" %>  
  
<html>  
    <head>  
        <title>Test Page</title>  
        <ww:head />  
    </head>  
  
    <body>  
        <vs:snapshot value="top.class" />  
        <ww:iterator value="list" status="status">  
            <li>  
                <ww:property value="top"/>  
                <vs:breakpoint active="true" name="personBP"/>  
            </li>  
        </ww:iterator>  
  
        <vs:breakpoint active="true" name="finalBP" value="#context"/>  
  
        <ww:iterator value="list" status="status">  
            <li><ww:property value="top"/></li>  
        </ww:iterator>  
    </body>  
</html>
```

Register the action and the result in xwork.xml

```
...
<action name="index" class="com.acme.test.TestAction">
    <result name="success" type="dispatcher">/WEB-INF/jsp/test.jsp</result>
</action>
...
```

Start the Switch if you already haven't done so (see Installation above)- it should start running on port 4281.

Visit the test action in your browser (preferably in another tab or window). It should render until you see about this:

- Rene (**BP PersonBP**)

It should keep loading (well, at least in FireFox). Now, when you check your Switch log you should see two connections - one from your Flash client, and one from the first breakpoint tag. If you can't see this, verify that the Switch is running, your Flash client is running (reconnect using the menu if necessary), and you're not dealing with a firewall or an already occupied port).



Exploring the stack

Let's examine the Flash client here for a moment. What you see is the result of a very first command sent by the breakpoint, to the Switch, which relayed it to the Flash client, which in turn transformed the XML into a tree.

We can see the name of the breakpoint (personBP), we can see the command (top), and we can see the result: a single object with a `toString()` method which resulted in 'Rene'. That makes sense, because if we look at our `TestAction`, we can see we've got a list of names, starting with Rene, and in the jsp we're iterating those names, and we have a breakpoint on every iteration. Now, we see "Rene" (normally followed by the classname between brackets, except when dealing with `java.lang.String` objects), and we can click on the arrow in front of "Rene" to make the tree expand, and show all the methods available to this object.



A full list will dropdown as soon as you click the arrow, and you can see every method, its return value and parameters. Now you can click every leaf in the tree, and the tool will create the OGNL expression for you. If you click on "Rene", **top** will display in the input field below. Click on hashCode(), toString(), .. and it will result in **top.hashCode()** and **top.toString()**. When you're dealing with getters, Maps or Collections/Arrays, the client will provide you with the correct expression.

Let's try issuing a custom command. Click on the input field below (besides the 'Query' button), and type **#context**, followed by a click on the Query button, or an <Enter>. A big list of objects will dropdown. These are all the objects which can be retrieved with the command **#context**. In fact, this is nothing more than a Map with a lot of Map.Entry objects in it. Once again, you can use the arrow to see what methods are available, but keep in mind: these are the methods for Map.Entry !



Once again, clicking on the object is enough to create the OGNL expression, so let's click "com.opensymphony.xwork.ActionContext.locale" [java.util.HashMap\$Entry], and in the input field below, you'll get **#context['com.opensymphony.xwork.ActionContext.locale']**. Click the Query button again, and you'll receive the answer: "en_US" [java.util.Locale] (or some other Locale). When you expand the node now, you'll see it has all the methods for java.util.Locale instead of the old Map.Entry methods ! For example, clicking the `getLanguage()` method will result in `#context['com.opensymphony.xwork.ActionContext.locale'].language`, and submitting that will get you "en" as a response, on which you can invoke more String methods, and so on, ad infinitum.

Now, this is nice, but there's more (of course, there's always more). There's a menu on top of the tree (Connection, Breakpoint, Stack, About), which we'll explore now. Connection is fairly obvious, it controls the connection to the Switch server, About just displays a mildly uninteresting popup, Breakpoint is something we'll talk about in a minute, which leaves us with .. Stack.

Stack simply contains a list of some common commands, including **#context** (which we used above), **#request**, **#session**, etc .. clicking on them will simply send the command and show the result. Please note that not all commands return something (some are just empty - for example, **#session** will be empty until you actually place something in it (yes, this tool does not bend reality nor logic).

Breakpoints

Now, let's go to the next breakpoint. Remember, this is the very first breakpoint, and we have placed a breakpoint inside a loop of a list with 5 names. Click the > arrow on the right of the Query button. Within an instant, you will be greeted by "Peter". Clicking the > arrow again will show the next entry, "Toby". Take a look again at the test page in our browser, and you'll see it actually has advanced 2 items as well. The Breakpoint menu on top has the very same command, plus some others: 'Next' (which we used now), 'Skip Next' (which will skip the next breakpoint) and 'Skip All' (which will skip all of the breakpoints and just render the page - surprise, surprise). Choose 'Skip Next' from the Breakpoint menu and see how it indeed skips one breakpoint, displaying "Patrick" rather than "Rainer" (take a look at the rendered page in your browser, you'll see there is no (BP personBP) entry next to Rainer's name).

More commands

Of course, you can do more than just simple object retrieval. How about testing ? You'll often find yourself using if/else structures, so we can easily test them as well. Let's try **top eq "Jason"**, and just like you expected (I hope, if not, back away from the computer), "false" will be the result of this statement (with a nice icon - thanks Eclipse !). Similary, **top eq "Patrick"** will return "true", as will **top.substring(0,2) == "Pa"**. Great !

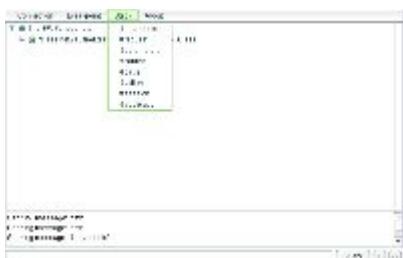
Use the next button (>) to proceed to our next breakpoint. You'll see the breakpoint name has changed to finalBP (check the jsp page, you'll see we used that name on our last breakpoint, outside the loop - if you can't find this in your jsp, you're one lousy copy/paster, and you should exchange your PC for a Mac immediatly), and we can see the #context result is being displayed again. This is because we actually specified value="#context" in our tag (it defaults to 'top' if it's not specified). Sending **top** again will display the Action class.

You can see it has a method getList(), so sending top.list will give us back our names list. Of course, nothing stops us from altering this list, so let's do that:

top.list.add("Claus"), press enter, and behold, a "true" tells us that we successfully added Claus to our list. A quick **top.list** indeed shows the name has been added. By the way, pressing the up key in the input field allows you to cycle through the used commands (a bit buggy atm though).

Snapshots

There is one last thing you might find interesting. There are snapshot tags that allow you to place anything on the stack (for comparison later on), which can be retrieved using #snapshots.



Closing thoughts

Since this was the last breakpoint, pressing next will cause the page to be fully rendered. When you take a look at it, you'll see that we have in fact our initial 5 names, wich visual breakpoint marks, and a secondary listing with the additional 'Claus' name added to it. When you reload the page, you'll get "Rene" again, and so on ..

I do suggest you take a look at the Tag reference; esp. the 'active' attribute is interesting (it allows for conditional breakpoints, thank you very much, and by default its value, when not specified, is equal to whether or not you have the WebWork development mode on. So keep this in mind.

Finally, I do realise that it's a bit awkward to use a Java daemon and a Flash client, and that we really should move to a single (preferably WebStart) Swing application instead. I just didn't have time for it (and I wanted to provide a quick popup in a webpage at first, so I guess one could say it's historically 'grown' 😊).

Tag reference

Breakpoint Tag

Name	Required	Default	Type	Description
name	false	net.vaultnet.explorer.Breakpoint.toString()	Object/String	The name you wish to give to this breakpoint.
active	false	webwork.devMode	Object/Boolean	Whether or not this tag is active. Accepts an expression.
value	false	top	Object/String	Initial request that will be executed against the OGNL stack.
host	false	localhost	String	The hostname where the Switch server is running on.
port	false	4281	int	The port the Switch server is running on.

Snapshot Tag

Name	Required	Default	Type	Description
value	false	top	Object/String	The expression result you want to store in this snapshot.

Tutorial

This page last changed on Mar 01, 2006 by [phil](#).



In Progress

This is a work in progress. Thank you for your patience.

1. [Getting Started](#)
2. [Create a webapp project structure for your web application](#)
3. [Lesson 1: Your first Action](#)
4. [Lesson 2: Putting some data in your Action](#)
5. [Understanding results](#)
6. [Understanding interceptors](#)
7. [Understanding IoC](#)
8. [Understanding validation](#)
9. [Understanding tag libraries](#)
10. [Portlet Tutorial](#)

Extra:

1. [Setting up Eclipse with Tomcat](#)

Old tutorials (these are out of date, but the concepts remain):

1. [Lesson 1 - Setting up webwork in a web application](#)
2. [Lesson 2 - An html form with no data](#)
3. [Lesson 3 - An html form with data](#)
4. [Lesson 4 - An html form with data, without getters or setters](#)
5. [Lesson 5 - Views](#) (JSP, Velocity, Freemarker)
6. [Lesson 6 - Interceptors](#)

Basic configuration and your first action - Hello WebWorld

This page last changed on Jan 03, 2007 by [phil](#).

Your first action - Hello WebWorld

TODO: finish this.

The Code

```
package com.acme.test;

import com.opensymphony.xwork.ActionSupport;

public class HelloWorld extends ActionSupport
{
    private String message;

    public String execute()
    {
        message = "Hello, WebWorld!";
        return SUCCESS;
    }

    public String getMessage()
    {
        return message;
    }
}
```

Let us create the view page. Create a file hello.jsp and place it under the YOUR_WEBAPP/WEB-INF/jsp directory:

```
<%@ taglib prefix="ww" uri="/webwork" %>
<html>
<head>
<title>Hello Page</title>
</head>
<body>
    WebWork says:
    <h1><ww:property value="message"/></h1>
</body>
</html>
```

Edit the WEB-INF/classes/xwork.xml file as shown below, adding the helloWorld action and 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 webwork defaults (from WebWork JAR). -->
    <include file="webwork-default.xml"/>

    <package name="default" extends="webwork-default">
        <default-interceptor-ref name="basicStack"/>

        <action name="helloWorld" class="com.acme.test.HelloWorld">
            <result name="success"/>/WEB-INF/jsp/hello.jsp</result>
        </action>
    </package>
```

```
</xwork>
```

Go ahead and try it now: go to the url http://localhost:8080/YOUR_WEBAPP/helloWorld.action 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.

- The WebWork filter (mapped on /*, meaning every request) receives the request for helloWorld.action.
- WebWork passes the request on to its ActionMapper, and lets this ActionMapper handle the request for a helloWorld action.
- The ActionMapper finds the class registered with the helloWorld name in the xwork.xml.
- By default, the execute() method gets called (but you can override this).
- execute() returns a String called SUCCESS, and WebWork looks again in xwork.xml to what and if any result is registered with that name. It finds the result (by default, a ServletDispatcher), which maps to /WEB-INF/jsp/hello.jsp.
- The page hello.jsp is processed (the <ww:property value="message" /> tag then uses OGNL to call the getter getMessage() of the HelloWorld.java action) and sent back to the browser.

Create a webapp project structure for your web application

This page last changed on Mar 22, 2006 by [mrdon](#).

Starting with a blank web application

You can use the `webapps/build.xml` file to create a project structure to help you get started quickly.
Just run 'ant new' and follow instructions.

```
new:
[echo]
[echo] +=====
[echo] |           -- Create a new web application --
[echo] +=====+
[echo]
[echo] [input] Enter the name of your new application [myapp]? ww-sample
[echo] [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
```

This task creates a new directory within the webapps dir.

For example, this is the setup for a new webapp project created with 'ant new' webapp name 'ww-sample':

```
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
```

You can now use the newly created project structure to get your webwork-based project

running.

You may want to look at the other sample webapp projects, to get some examples of advanced usages. Have a look at the showcase, starter, or shopping-cart applications.

CRUD Demo I

This page last changed on Jul 16, 2006 by [phil](#).

WebWork CRUD Example

(Last update: 04/28/2006)



This is a copy of the tutorial written for <http://www.learntechnology.net>

Introduction

Welcome to the WebWork (WW) CRUD Example. This example was created to be as simple as possible and outline the differences with the other CRUD example on this site (Apache Struts 1.x, JSF, ..), and as such, it does not use all of the advanced (integration) features such as Spring IoC, Hibernate Open-session-in-view, OS Sitemesh, annotations, etc .. For these and other examples, please refer to the official WW site.

Note: Larry has also created a second WW tutorial, which uses Spring and JSTL - so you might want to check that one out as well.

WebWork

WebWork is a traditional MVC2 action-based framework (such as Struts, Stripes, Simple, ..) as opposed to the newer event-based frameworks (such as JSF, Wicket, Rife, ..). WebWork uses XWork under the hood, a command-pattern based framework that handles conversion, validation, interception, and a lot more. WebWork was originally started as an effort to overcome the problems of Apache Struts 1, and has recently agreed to join the Apache Foundation again to become the successor of Struts 1, consequently named Struts Action 2 ('Action' because Struts has been divided in a number of standalone subprojects with different technologies).

Therefore, besides a change in the package names, this example will be entirely valid for Struts Action 2.

.war layout

The .war file you can download on this site can be dropped in your servlet container and contains the source code under the WEB-INF directory. The .war layout is also kept as simple as possible:

```
- learn-ww
-- [css] (contains the stylesheets)
-- [WEB-INF]
---- [classes] (contains the compiled src files)
---- [lib] (contains the dependencies)
---- [jsp] (contains the view pages)
---- [src] (contains the source files)
---- web.xml (our webapplication descriptor)
-- index.html (simple redirect page)
```

Configuration files

WEB-INF/web.xml

The webapplication's descriptor file contains one filter and its mapping. By default, the filter is mapped to `/*`, meaning all requests will be intercepted, but only those ending with a specific suffix (`.action`, by default) and certain special paths (for static files) will be processed and handled by WebWork.</p>

```
<web-app>
    <display-name>WebWork Demo</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>
</web-app>
```

More information on [web.xml](#)

WEB-INF/classes/xwork.xml

xwork.xml contains the configuration for XWork: actions, validation, interceptors and results are defined in there.

To understand these terms, we'll need to take a look at WebWork's (and XWork's) architecture. A basic request goes a bit like this:

A request is submitted, and our ActionMapper will try to determine the correct Action to execute. To find it, it will look up the registered names in your xwork.xml. If one is found, before executing the Action, it will loop through a defined stack of interceptors.

Interceptors are a very important part of WW - they will be invoked before and after your action is executed, and as such, they are perfect for validation, authentication, open-session-in-view patterns, catching exceptions, setting or altering parameters, hiding complex operations, and more. WW provides a number of prebuilt stacks with a ranging number of features, but nothing keeps you from defining your own interceptor stack with custom interceptors.

One of the most practical interceptors is the 'params' interceptor. It will translate your request parameters to set them on your action. Thus, if your action had a `setName(String)`, and one of your request parameters is called 'name', then WW will set the name for you. Not so special, you say ? Ok, how about `setId(Long id)` ? This will work just fine as long as your id parameter value can be converted to a Long. Still not special enough ? How about submitting a parameter named `employee.id` ? As you might have guessed, this will invoke a `getEmployee().setId(Long id)`. WW handles all common objects (Integer, String, Date, .. and Arrays, Lists, Maps, ..) And for those who just can't get enough, you can always add your own convertors for those (or your own) very complex objects (think social security number).

An important aspect of a framework is validation. Adding validation can be a slow and complex process - not to mention the user feedback when something goes wrong. As we think you shouldn't reinvent the wheel, so WW has a loosely coupled validation framework which you'll see in action when you try to insert or update an Employee.

Now, when an Action is executed, the result will be used to control the flow - these simple Strings ("success", "error", "input", ..) will be used to invoke a certain Result - this Result can be a dispatcher to a jsp file, render a freemarker template, generate a chart, output xml, you name it. And, it's totally independend from your Action. Note when validation fails, the result will be by default "input".

Now, as soon as the result is rendered/dispatched/executed/.. WW will loop through the interceptors again in reverse order - which is perfect for cleaning up resources, logging, timing, .. etc.

Let us take a more detailed look at our [xwork.xml](#):

```
<xwork>
    <!-- Include webwork default (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="paramsPrepareParamsStack"/>
        <action name="index" class="net.vaultnet.learn.action.EmployeeAction"
method="list">
            <result name="success"/>/jsp/employees.jsp</result>
            <!-- we don't need the full stack here -->
            <interceptor-ref name="basicStack"/>
        </action>
        <action name="crud" class="net.vaultnet.learn.action.EmployeeAction"
method="input">
            <result name="success" type="redirect-action">index</result>
            <result name="input"/>/jsp/employeeForm.jsp</result>
            <result name="error"/>/jsp/error.jsp</result>
        </action>
    </package>
</xwork>
```

First thing you encounter is the inclusion of the webwork-default.xml. This file, located in the root of your webwork.jar, contains the default interceptors, interceptor stacks and results so you don't have to register them.

Next up are the packages - packages can inherit from other packages - and are used to group actions together on different namespaces. Not important right now, so we are going to use the default one. We also specified we want to use the paramsPrepareParams interceptor stack, for reasons explained later. Two actions are defined, and they both use the same Action class, EmployeeAction. One is registered with the name 'index', and will be used for the index page, while the other one, 'crud' will be used to execute the various create/read/update/delete actions. You also note they list different method attributes: one will execute, by default, the list method while our crud action goes with the input method.

After that, we arrive at the results. Remember how I told each Action execution has to return a String result ? Well, here they are. For our index action, we don't require any input, we will assume nothing goes wrong, so we only list the success result (Note: you can register global results as well). This result uses, under the hood, the default DispatcherResult result-type to dispatch the request to an employees.jsp file. Also note the fact that since I didn't need the full stack (no validation, fileupload, preparing, or other funky things), I chose to use the basic interceptor stack for this request.

It gets more interesting for our crud Action: besides the success result, we also specify the input result (which dispatches to our input form) and the error result (which is returned when an exception is thrown during the Action execution - for example a database exception). You can also see we specified a different result-type for our success result, in casu the 'redirect-action'. This is nothing more than a fancy redirect which will append the chosen WW suffix, so we could have also used the redirect result-type, with index.action as its text. But this approach is slightly better since it is suffix agnostic (you can switch suffixes very easily in WW, so that is why we always advise to program suffix agnostic and let the

framework handle it for you).

More information on [xwork.xml](#)

WEB-INF/classes/webwork.properties

This file only contains one line, and is used to set webwork specific settings (such as which IoC container to use, which fileuploader, what templates, etc ..). You don't really need it, but for i18n reasons, we use it to register our resource bundle 'guest.properties'.

More information on [webwork.properties](#)

WEB-INF/classes/guest.properties

This guest.properties file will contain the keys and values to internationalize your webapplication in a straightforward way. Rather than hardcoding Welcome ! in your page, you should use specify a key (eg. 'welcome_msg') with a certain value ('Welcome !'). By adding new resource bundles, you can override the key and specify a different value (eg. 'Bienvenue !' for a french Locale).

```
webwork.custom.i18n.resources=guest
```

The code

Since 90% of the code is identical to the other CRUD examples displayed on this site, we'll only analyze the Action class, EmployeeAction.

As always, first things first - the class definition:

```
...
import com.opensymphony.xwork.Action;
import com.opensymphony.xwork.ActionSupport;
import com.opensymphony.xwork.Preparable;
...

public class EmployeeAction extends ActionSupport implements Preparable {
    ...
```

Now, first there is the extending of the ActionSupport class - although you don't have to extend it, it provides a lot of useful extras, so you are encouraged to extend and override parts of it, but you don't have to. The interface we are implementing is a bit more interesting. The Preparable interface only defines one method, public void prepare(). By implementing this interface in your Action, you tell the prepare interceptor to call this method on your Action - so that makes it perfect to, for example, retrieve objects from a database, which is what we're 'faking' here by requesting an Employee object from the EmployeeService.

There are quite some different interfaces that you can implement that will be used by interceptors (SessionAware, ServletAware, ..), but you need to be sure the interceptors are listed in your interceptor stack, or the interceptor won't be executed on your action.

Besides the prepare() method we just explained, our Action also contains the doInput(), doSave(), doList() and doDelete() methods. Remember how we specified the method attribute in the xwork.xml file ? Well, these are the methods that are going to be executed. Technically, you could have defined input(), save(), list() and delete() as well, but WW will automagically find the correct method. Originally, this was because often a default() method is declared - but since default is a reserved keyword, you have to make that doDefault(). You don't have to care about that, but just so you know.

Let's explore the doList method, which is the default method we specified in our xwork.xml for our index action.

```
public String doList() {
    employees = empService.getAllEmployees();
    return Action.SUCCESS;
}
```

Surprisingly simple, no ? Simply fill in the employees object, and return "success" (defined as a final static variable SUCCESS). That's it ? That's it. The only thing you need now to show the list in your view layer is a simple getter for employees in your Action. The reason for this is another WebWork feature: the ValueStack.

The ValueStack: it's magic, baby !

The ValueStack is like a normal stack. You can put objects on it, remove them from it, and query it. And even more, you can query it with expressions ! It's the heart of WebWork, and allows easy access to a wide range of objects from nearly any place in the framework: interceptors, results, tags, .. you name it. Now, when you execute your Action, it will be placed on top of the stack.

So, once it's on the valueStack, you could query for it 'employees' - a special glue language called OGNL will then transform that request to 'top.getEmployees()' - and since top is the top of your stack, and your Action is located just there, it will invoke the getEmployees() method on your Action, and return the result.

More information on [the WebWork architecture](#), [Interceptors](#), [OGNL](#)

First view: the employee listing

WW defines a lot of tags that will dramatically speed up your development. They can alter or query the value stack, render input forms, javascript widgets, loop iterable objects, and so on. On top of that, they have different themes, which add even more functionality/layout by the switch of a parameter. Themes are out of the scope of this example, but you should definitely check them out.

```
<!-- /webwork is automatically found, you don't have to register it in your web.xml -->
<%@taglib prefix="ww" uri="/webwork" %>

<head>
    <link href="
```

I'll explain the first two tags: ww:url and ww:text (ww: being the most commonly used taglib prefix - and

it helps to clear the confusion when talking about html tags).

The `ww:url` tag allows you to build urls with parameters for actions (rather than having to type them out manually), and the `ww:text` tag will look up keys in resource bundles in the `valueStack` (depending on the locale). So adding a new language would be a breeze. In these cases we build an url '/css/main' and we display the value of a key 'label.employees' from our `guest.properties` resource bundle.

```
<ww:url id="url" action="crud!input" />
<a href="<ww:property value="#url"/>">Add New Employee</a>
```

There's something special about this as well. We already know what the `ww:url` tag does, but here we specified an `id`, and our `action` attribute looks really weird.

First, about the `action` value, forget about that for now - we'll get to that in a minute.

Then there's that `id` attribute. A lot of tags (esp. those that generate/display something) share it, and it allows you to store the result of the tag on the `valueStack`, as `#(id_value)`. Thus, my generated url would be available on the stack as `#url`, and as you can see, we use that same `#url` to output it later on in our anchor tag.

More information on `the WebWork tags</p>`

The ActionMapper

The WW ActionMapper doesn't just map names to Actions - it can also change the method, or even action, by using special requests. The weird value we encountered above, `crud!input`, tells the ActionMapper to invoke the `input()/doInput()` method of the Action known as `crud`. So for example, you could have several slightly different methods, and rather than having to register each of them in the `xwork.xml` file, you can use the `!` notation to specify which method to execute.

Another thing is the fact that you can override which action/method to invoke based on a special `name:action` parameter, which we'll use later on in our `employeeForm.jsp` to make a nice cancel button.

After this small intermezzo, back to the employee listing.

More information on the [ActionMapper](#)

More tags: `ww:iterator`, `ww:if/else`, and `ww:property`

Take a look at the following lines from our `employees.jsp`:

```
<ww:iterator value="employees" status="status">
  <tr class="<ww:if test="#status.even">even</ww:if><ww:else>odd</ww:else>">
```

The `ww:iterator` tag does pretty much what you expect from it: it loops through an array/list/collection, and pushes each object it encounters on the stack on each iteration. It also has a helper attribute called `status`, which generates a `status` object that keep track of the index, tells you if you're in an even or odd row, and so on, while you're inside the `ww:iterator` tag.

As you can see, we use that same status object in our next tag, the `ww:if` and its compagno, the `ww:else` tag. The test attribute of the if tag allows you to query the valueStack with an OGNL expression, which is exactly what we do here: see if the status object from our `ww:iterator` tag we put on the valueStack, returns true when the method `isEven()` is invoked. The corresponding `ww:else` method is then executed if the test method return false.

Finally, there's the `ww:property` tag, which is used for displaying objects/expressions from the ValueStack. Examine this:

```
<ww:property value="age" />
```

This might be a little confusing at first, but it's very simple. This actually translates to `top.getAge()`, meaning we'll execute a `getAge()` on the top object on our ValueStack. Which happens to be .. the Action ? Nope. The employees List ? Closer. The current Employee object in the employee list ? Bingo.

Why ? Because I told you what the `ww:iterator` tag did: it places each object on the top of the stack. Since the employees List contains Employee objects, an Employee object gets 'on top'. So that's why you can simply type `age` (or `top.age`) and have the employee's age printed out.

Adding & Editing an employee

Easy enough: return "input" from your Action, and register the result in your `xwork.xml` with a dispatcher to `employeeForm.jsp`.

```
public String doInput() {
    return Action.INPUT;
}
```

Nothing to it. So let's take a look at how we are going to edit an Employee.

Editting, or better, preparing for editing, comes in many flavours. Fact is, you often need 'extra' objects, like a list of possible departments in our case, to be set up before rendering the insert or edit page. In WW, there are quite some ways to accomplish this, but two approaches are recommended:

- Using a `prepare()` method and interceptor to setup any additional objects you need
- Using a `ww:action` tag to use another action to create the objects for you - for example, a `DepartmentAction` could return a list of Department objects.

Here I'll show you both, but in the example application you'll only find the first one (might change, that's why I'm adding it here).

The prepare approach

A little rehearsal: the `prepare` interceptor, when listed in your interceptor stack, will call the `prepare` method. We use this `prepare` method to set up our deparments, so the list will be available whenever our crud action is called.

We also use this to retrieve an employee bean whenever an id is set - which is precisely how we make the difference between inserting and editing (and we use a similar method in the `doSave()` method in our `EmployeeAction`). So, visiting the `crud!input` action without an `employee.id` parameter to retrieve the employee, will result in an empty form, while passing the parameter will result in an Employee object to be retrieved for editing.

More information on the [Prepare interceptor](#)

The **ww:action** approach

The other commonly used approach is to use the **ww:action** tag. The **ww:action** tag allows you to execute (additional) WW actions, which makes them perfect to generate objects for input forms, such as select boxes. You could create for example, a Department action, which does nothing more than listing an `doList()` method to set up a list of departments, and getter for it, called `getDepartments()`.

Register the `DepartmentAction` in your `xwork.xml` as 'department', and you can simply call it in your page, and use the id approach we used in the `ww:url` tag to store the result on the stack under a custom name called `allDepartments`:

```
<ww:action name="department!list" id="allDepartments"/>
```

Now, this, by itself, does not do much. I'll show you later when we talk about the `employeeForm` page how to use it as an alternative approach.

More information on the [ww:action](#) tag

Forms made easy

The `employeeForm.jsp` page is really concise. I told you about the themes that WW uses under the cover, right ? Well, those are going to be responsible for rendering our form, complete with labels for names and errors, input fields, and so on.

But first, we find another useful tag: `ww:set`

```
<ww:if test="employee==null || employee.employeeId == null">
    <ww:set name="title" value="%{'Add new employee'}"/>
</ww:if>
<ww:else>
    <ww:set name="title" value="%{'Update employee'}"/>
</ww:else>
```

The `set` tag allows you to store certain objects on the stack (as well as their scope - `request/session/page/..`), which is what we're going to do here because out of sheer laziness (and performance reasons) I refuse to do the same `if/else` more than once. As you can guess, it relies on the same principle as the `id` attribute of the `ww:url` tag we saw earlier, meaning I can access it with `#title` on the `ValueStack`.

More information on the [ww:if](#) tag, [ww:else](#) tag

The actual form

```
<ww:form action="crud!save.action" method="post">
```

```

<ww:form name="employee" value="#{employee}">
    <ww:textfield name="employee.firstName" value="#{employee.firstName}" label="#{getText('label.firstName')}" size="40"/>
    <ww:textfield name="employee.lastName" value="#{employee.lastName}" label="#{getText('label.lastName')}" size="40"/>
    <ww:textfield name="employee.age" value="#{employee.age}" label="#{getText('label.age')}" size="20"/>
    <ww:select name="employee.department.departmentId" value="#{employee.department.departmentId}" list="departments" listKey="departmentId" listValue="name"/>
    <ww:hidden name="employee.employeeId" value="#{employee.employeeId}" />
    <ww:submit value="#{getText('button.label.submit')}" />
    <ww:submit value="#{getText('button.label.cancel')}" name="redirect-action:index" />
</ww:form>

```

Wow - a lot of code at once. Let's dissect it tag by tag. It may seem complicated, but as you'll see, it's actually really easy.

The `ww:form` tag generates a standard html form (note: there's a small bug when using the `action!method` notation without suffix, that's why I am including `.action` here), while the first `ww:textfield` will generate an `<input type="text" ... />` - but wait, it does more. It transforms your first tag from this:

```

<ww:textfield name="employee.firstName" value="#{employee.firstName}" label="#{getText('label.firstName')}" size="40"/>

```

Into this:

```

<tr>
    <td class="tdLabel">
        <label for="crud!save_employee(firstName)" class="label">First Name:</label>
    </td>
    <td>
        <input type="text" name="employee.firstName" size="40" value="" id="crud!save_employee(firstName)"/>
    </td>
</tr>

```

Let's analyze that `ww:textfield` tag in greater detail. First thing we encounter is the `name` attribute, which is similar to the HTML `input` tag's `name` attribute. So it is by itself, not very special. However, as we've seen above, this allows WW to call the `getEmployee().setFirstName()` method - and even, if necessary, create the `Employee` object for you. So no more tens of property setters in your action - just one setter for a good ol' POJO, and you can call its setters right away (Note: for those who are concerned about malicious injections, you can limit what can be set on your POJO) !

The `value` attribute uses a special `%{..}` notation, to indicate the value is not just a string '`employee.firstName`', but in fact an expression that should be looked up on the ValueStack. So this one will make OGNL analyze the content, and look up on the ValueStack to see if it can find a method named `getEmployee()`, which returns a POJO which has a `getFirstName()` method on it. Now, when such an expression returns null (since our `Employee` pojo is NOT initialised - we are doing an insert here, remember ?), WW creates the `Employee` object for you, and its `getFirstName()` returns, of course, null. So, we're actually cheating here, because it will allow us to reuse that same form when we are going to edit an `Employee`. In that case, our `getEmployee()` would return an initialised object, so its `getFirstName()` would return a value, and thus display it in our input field ! Great - reusal of forms is always nice (of course, you can always use two seperate forms, but you don't need to).

Ok, so `%{..}` indicates an expression on the ValueStack, and returns a blank if a null is returned, or the `toString()` value otherwise. The OGNL analyzer is pretty powerful, so you can do things like

```
value="my_special_valentine_is_%{girlfriend.name}", %{100 * loan.tax}, or even %{new int[100]}  
and %{new java.util.Date()}.}
```

Now, we saw earlier that we could use the `ww:text` tag to retrieve values from resource bundles for i18n reasons. Now that begs the question, how do we use those same values in our tags ? Let's say we want to i18n'ize our textfield label. Something like this ?:

```
<ww:textfield name="employee.firstName" value="%{employee.firstName}" label="<ww:text  
name="label.firstName"/>" size="40"/>
```

Ugh. No, that's ugly, and it wouldn't work either. The solution is much cleaner and simpler: use another expression ! If you were to check the extra methods provided by making our `EmployeeAction` extends `ActionSupport`, you would see a method called `getText(String key)`. This method will look up a value in the resource bundle by its key, which is exactly what we need. So, the label would become something like this: `%{getText('our_key')}`. Makes sense ? Thought so.

Meet the next tag, `ww:select`, which renders a select box using an iterable collection:

```
<ww:select name="employee.department.departmentId" value="%{employee.department.departmentId}"  
list="departments" listKey="departmentId" listValue="name"/>
```

We already covered the `name` attribute, so let's look at the `list` attribute: `departments`. Hmm, this might seems strange at first. Where does it come from ? Well, perhaps it makes much more sense to see this: `list="%{departments}"`. Yes, it is in fact an expression on the ValueStack that will query our action for the departments we've setup before ! Then, why are we missing the `%{..}` ? The answer is twofold: you can still write `%{departments}`, and it would work as you expected. But you have to realize, that whatever you are going to iterate is going to be an expression ! So, being pretty lazy and with all this Ruby-on-rails 'minimalistic' code, we decided we might as well save you a couple of RSI-related finger moves and let you discard the `%{..}`.

Another quick intermezzo: you can use expressions to make your own list in an expression - which is perfect for small yes/no and male/female/eunuch selections - like this:

```
<ww:select name="gender" list="#{'male':'Male', 'female':'Female'}" />
```

By the way, did you remember the `ww:action` tag we used as an alternative to the `prepare` method to fill up the select box ? We can now use the action we executed and stored on the ValueStack by referencing it by its id:

```
<ww:select name="gender" list="#{allDepartments.departments}" />
```

So, instead of getting the departments from the current action, we used the `ww:tag` to execute another one, and now we're using that action to get the department List. This allows you to split up, and reuse Actions (Note: you can go even further and program your own components with built-in actions, but that's out of scope).

Let's get back to the original plan.

Now, the `listKey` and `listValue` attributes tell WW what it should use as keys and values in our select box -

and what do you know, those (hidden) expressions are going to be invoked on each object it encounters in your list - in this case, `getDepartmentId()` and `getName()`.

Finally, the value attribute will tell WW where to place the typical 'selected' attribute in our generated options, and it will print it as soon as the value expression equals the key expression. Thus, in our case, as soon as `your.employee.getDepartment().getDepartmentId()` equals the `getDepartmentId()` expression on one of the Department objects in our departments List. Since we don't have an employee we're editing, the expression would return null, so no selected attribute would be printed.

Finally, the last two tags, the submission tags. Submission tags, yes. One for submitting the form, and another one to cancel it. The first submit button, submits the form to the form's action attribute, in case crud!save.

```
<ww:submit value="${getText('button.label.submit')}" />
<ww:submit value="${getText('button.label.cancel')}" name="redirect-action:index" />
```

The second one is more interesting though. It also submits the form to the same crud!save action as the first one, but lists a special name attribute. This name attribute will cause the ActionMapper to intercept the call, and in this case, redirect it to another action. No more fiddling with javascript to have forms with multiple submit buttons - it's all done for you, without any javascript or the troubles that come with it.

More information about [themes and templates](http://wiki.opensymphony.com/display/WW/Themes+and+Templates), [form tags](http://wiki.opensymphony.com/display/WW/Form+Tags), [action mapping](http://wiki.opensymphony.com/display/WW/ActionMapper)

Validation

Alright, so you've set up the form for adding and updating employees. This is the point where you normally start sweating, you hands start shaking and you feel slightly dizzy. Have no fear, WW is here !

WW uses XWork's validation framework internally. It allows you to validate Models and Actions using a set of specialised (field)validators, grouped together in an xml file named `YourModel-validation.xml` or `ActionName-validation` (and, in the case of the alias, `ActionName-alias-validation.xml`). Since we only want to validate the crud action, we create a file `EmployeeAction-crud-validation.xml` and place it in our classpath (mostly next to our compiled Action class).

```
<validators>
    <field name="employee.firstName">
        <field-validator type="requiredstring">
            <message key="errors.required.firstname" />
        </field-validator>
    </field>
    <field name="employee.lastName">
        <field-validator type="requiredstring">
            <message key="errors.required.lastname" />
        </field-validator>
    </field>
    <field name="employee.age">
        <field-validator type="required" short-circuit="true">
            <message key="errors.required.age" />
        </field-validator>
        <field-validator type="int">
            <param name="min">18</param>
            <param name="max">65</param>
        </field-validator>
    </field>
</validators>
```

```

<message key="errors.required.age.limit"/>
</field-validator>
</field>
</validators>

```

A very important reminder: validation is once again done by an interceptor, so it should be in your stack. Even more important, validators 'query your Action/Model' and NOT your request ! Keep this in mind at all times.

With that out of the way, let's analyze this snippet: there are two types of validators: field validators and general validators. We start by analyzing the field employee.firstName - so this means: we will analyze the result from the invocation of getEmployee().getFirstName() on our Action, not the request parameter named employee.firstName !

Field validators will not validate input fields - they are named field validators because they will automatically mark the input field in your form with the validation error, whereas normal validators would create actionErrors.

So, first we apply a requiredstring validator to the getEmployee().getFirstName() return value. There are quite a few validators, ranging from requiredstring, required, intrange, emailaddress, url, .. etc. Writing your own validator is not hard, and it can be reused easily. Here we use the requiredstring, which does 2 checks:

- check the availability of the string (!= null)
- check the length of the string to be > 0

Why the stringlength greater than 0 ? Well, if you are submitting an empty text input field, the variable name will be send, and thus will WW set our field name to an empty "" String (contrary to not sending a parameter normally results in nothing being set, or null).

The employee.lastName validation is exactly the same as the firstName. The only difference is in the validation of the age property.

We use in fact 2 validators: a 'required' one, which will make sure that our age is in fact set, and not null, and another one that will check the range to make sure our employee is between 18 and 65 years old. If course, if the first validator fails, we shouldn't continue processing, so that's why we can specify the short-circuit attribute. If a validator short-circuits the validation, validation is failed and skips to the current (field)validator.

The "int" validator takes 2 optional parameters: min and max, who can be set by providing two simple param tags (most items in WW can be configured that way). This way, we ensure validation to be loosely coupled with our code and don't require re-compilation.

Of course, you should also be able to i18n'ize your error messages, so that's why we providing a <message key="my_key"/> - if you prefer otherwise, you can always add a hardcoded text as a child element of the message tags.

It doesn't stop there. The text you pass, be it hardcoded or a looked-up value, can in fact contain OGNL expressions as well !

Take a look at the value we get back from the 'errors.required.age.limit'-key:

```
errors.required.age.limit=Please provide an age between ${min} and ${max}.
```

And guess what: min and max are indeed the two parameters we just set before ! Using OGNL expressions you can retrieve whatever you want from your action, giving you really nice error messages (we like nice error messages). In fact, you can not only use these expressions in your error messages, but you can even set the min and max parameters dynamically. Different types of employees could have different age requirements - OGNL and polymorphy to the rescue !

More information about [Validation](#)

Conclusion

This 'quick intro' turned out a bit longer than I anticipated. We haven't even barely touched the possibilities with WebWork - different templates, IoC (with XWork or Spring), annotations, REST-ful action mappings, components, Hibernate integration, ajax support, pdf/xml/rss/.. generation, groovy, More info on 3th party integration.

When you look at the various frameworks out there, they all let you create a certain type of (web)application really, really quickly. It's only when you want to add new/different/complex things, that you discover the limitations of the framework, often meaning serious hacking to get it working. WebWork is different. It is no out-of-the-box framework, where you just click a few buttons to generate a blog/cms/product catalog, but it's a framework in the true spirit of the word. Its architecture and design is so flexible that we yet have to discover where we cannot use it for, and as such it's a great overall framework that should belong in the backpack of any serious Java (web)application programmer.

Examples

This page last changed on Mar 10, 2006 by [phil](#).

- [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 Apr 19, 2005 by [jcarreira](#).

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](#)

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](#)

Chat Application

This page last changed on Jun 18, 2004 by [plightbo](#).

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 15, 2006 by [dwangel](#).

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 href="<%= request.getContextPath() %>/logout.action">Logout</a>
<br />
<a href="<%= 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 getPassword() {
        return passwd;
    }

    public void setPassword(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>
```

Getting Started

This page last changed on Mar 23, 2006 by [rgielen](#).

Preface

WebWork is a popular, easy-to-use MVC framework. For more information on the WebWork project, please visit the [WebWork Project Home](#).

This document will help you to get started with WebWork, enabling you to run the examples and demonstrations provided even if you are not an experienced Java web application developer. Nevertheless, WebWork is geared towards developers that have an understanding towards certain technologies. Before diving deeper into how Webwork works, it is recommended that you review the concepts below:

- Java
- Servlets, JSP, and Tag Libraries
- JavaBeans
- HTML and HTTP
- Web Containers (ex. Tomcat)
- XML

Distribution Download and Information Resources

Download Webwork Distribution

You can download the full WebWork distribution [here](#). It contains the webwork.jar file as well as full documentation, sources, all [required and optional dependent libraries](#), and examples. For more information on how to build WebWork from source or even from a clean CVS checkout, please refer to [Building Webwork](#).

Online Information and Project Resources

There are many online sources of information, support and project resources around. Here are links to help you find your way:

- [Download Webwork](#) - download Webwork Distribution
- [Join the WebWork Forums](#) - The forums are full of active developers, contributors, and power users - often even available for chat. This is the best and quickest way to get a question answered.
- [Subscribe to Mailing List or browse Mail Archive](#) or [post a question](#). 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](#) - Browse CVS and source at java.net
- [Webwork Wiki](#) - Powered by [Confluence](#), the professional J2EE wiki

- [Webwork Bugs & Issues](#) - Powered by [JIRA:Bug & Issue Traking System](#)
- [OpenSymphony Home](#)

Distribution Quickstart

Overview

The distribution contains the following directory layout:

```
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
```

The docs directory contains the current Javadocs, the full user documentation including the document you are reading, and taglib documentation, as well as Clover, JUnit and dependency reports for the build.

The dist directory contains WebWork files with different packaging:

- webwork-nostatic-<version>.jar: containing only WebWork without the static content
- webwork-static-<version>.zip: containing the required WebWork static dependencies

The lib directory contains the required as well as optional [Dependencies](#) for Webwork, 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
```

Note that none of the optional packages are required to use Webwork. If you wish to use certain features such as JasperReports or Java 5 (Tiger) generics and annotation support, you must include the optional packages.

Webwork also comes packaged with all the source files and templates for the JSP tags.

Running demos with QuickStart

WebWork provides a quick way to get started called QuickStart. QuickStart is essentially a combination of a few technologies, 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 portlet 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.

Your first WebWork Application

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 minimum 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>

```

```
</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](#)

Installation

This page last changed on Dec 21, 2006 by [phil](#).

Installation

Installation of WebWork is not very different from any other framework. [Download](#) the latest release (or nightly build, if you like living on the edge), and unpack the compressed file. You should see the following directory structure (left out some files for brevity):

- [webwork-2.2.x]
 - [dist]
 - webwork-static-2.2.x.zip //contains static (javascript) resources such as dojo, jscalendar, tooltips, ..
 - webwork-nostatic-2.2.x.jar //smaller webwork jar, contains no static resources
 - [docs] //wiki documentation in html and pdf
 - [lib] //jars in subdirectories required for a webwork installation
 - [src] //full source from webwork unpacked
 - [webapps] //example webapps
 - webwork-2.2.x.jar //webwork framework
 - webwork-src-2.2.x.jar //webwork framework source code

Webapp layout

Now, a typical (minimal) layout for a webapplication in WebWork may look like this:

- [js] //contains extra javascript files
- [images]
- [template] //contains extra templates for custom themes, etc ..
- [WEB-INF]
 - [classes] //contains your compiled application
 - [xwork.xml](#) //contains your action declarations
 - [src] //contains your application's source files
 - [lib] //contains the required libraries
 - [pages] //contains the view layer (jsp, freemarker, velocity)
 - [applicationContext.xml](#) //required by the Spring IoC container
 - [web.xml](#) //standard webapp descriptor

Minimal dependencies

The following libraries should be copied to your WEB-INF/lib folder (they can be found in the webwork/lib folder in various subdirectories).

If you get a `ClassNotFoundException`, it mostly means that you should add a jar from the webwork/lib folder. If you're not sure about which jar, use a standard decompression tool (7-zip, winzip, ..) to open the files and look inside to see if you can find the correct .class file.

- cglib.jar

- commons-attributes.jar
- commons-logging.jar
- freemarker.jar
- ognl.jar
- oscore.jar
- rife-continuations
- spring-aop.jar
- spring-beans.jar
- spring-context.jar
- spring-core.jar
- spring-web.jar
- webwork-2.2.x.jar
- xwork.jar

Further reading

With this setup, it's time to move onward. First, you should edit your [web.xml](#), then your [xwork.xml](#). Have a look at the [Spring](#) integration (you'll have to make sure your applicationContext.xml is correct), and finally move on to the first [Tutorial](#).

QuickStart

This page last changed on Jan 11, 2007 by [tm_jee](#).

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 lets you tweak how the deployment happens and how it is

configured as much as you'd like. Consider the sample quickstart.xml file:

An error occurred: <http://svn.opensymphony.com/svn/webwork/webapps/showcase/quickstart.xml>. The system administrator has been notified.

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.

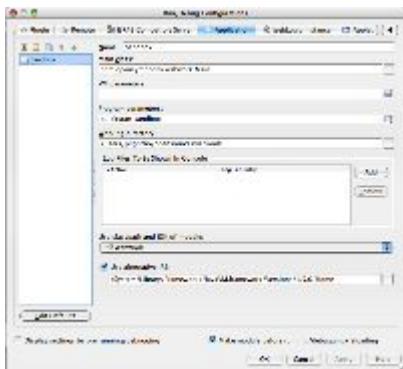
The Jetty server's context ClassLoader is specified as a custom ClassLoader that looks at the source files in **webapps/xxx/src/java** and compiles them on the fly. These classes are also reloaded whenever a change is detected.

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 its 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.

Lesson 1 - Setting up webwork in a web application

This page last changed on Sep 06, 2005 by [jcarreira](#).

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](#)

Lesson 2 - An html form with no data

This page last changed on Mar 08, 2005 by [plightbo](#).

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 helloWebWorld.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 execute() 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](#)

Lesson 3 - An html form with data

This page last changed on Feb 04, 2005 by [plightbo](#).

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](#)

Lesson 4 - An html form with data, without getters or setters

This page last changed on Feb 06, 2005 by [plightbo](#).

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 Jun 12, 2006 by [john2ee](#).

Index

1. [Introduction](#)
2. [Installing Eclipse](#)
3. [Installing JBoss Portal 2.2](#)
4. [Creating the project](#)
5. [Classpath settings](#)
6. [portlet.xml](#)
7. [web.xml](#)
8. [Hello World!](#)
9. [xwork.xml](#)
10. [JBoss Portal descriptors](#)
11. [Deployment](#)
12. [Next step](#)
13. [Re-deployment](#)

Step-by-Step Tutorial

Introduction

This tutorial walks you through the process of building a simple portlet application, using Eclipse, JBoss Portal 2.2 and the WebWork Portlet framework.

Installing Eclipse

In the tutorial, we will be using Eclipse 3.1.1 which can be downloaded from <http://www.eclipse.org>

Installing JBoss Portal 2.2

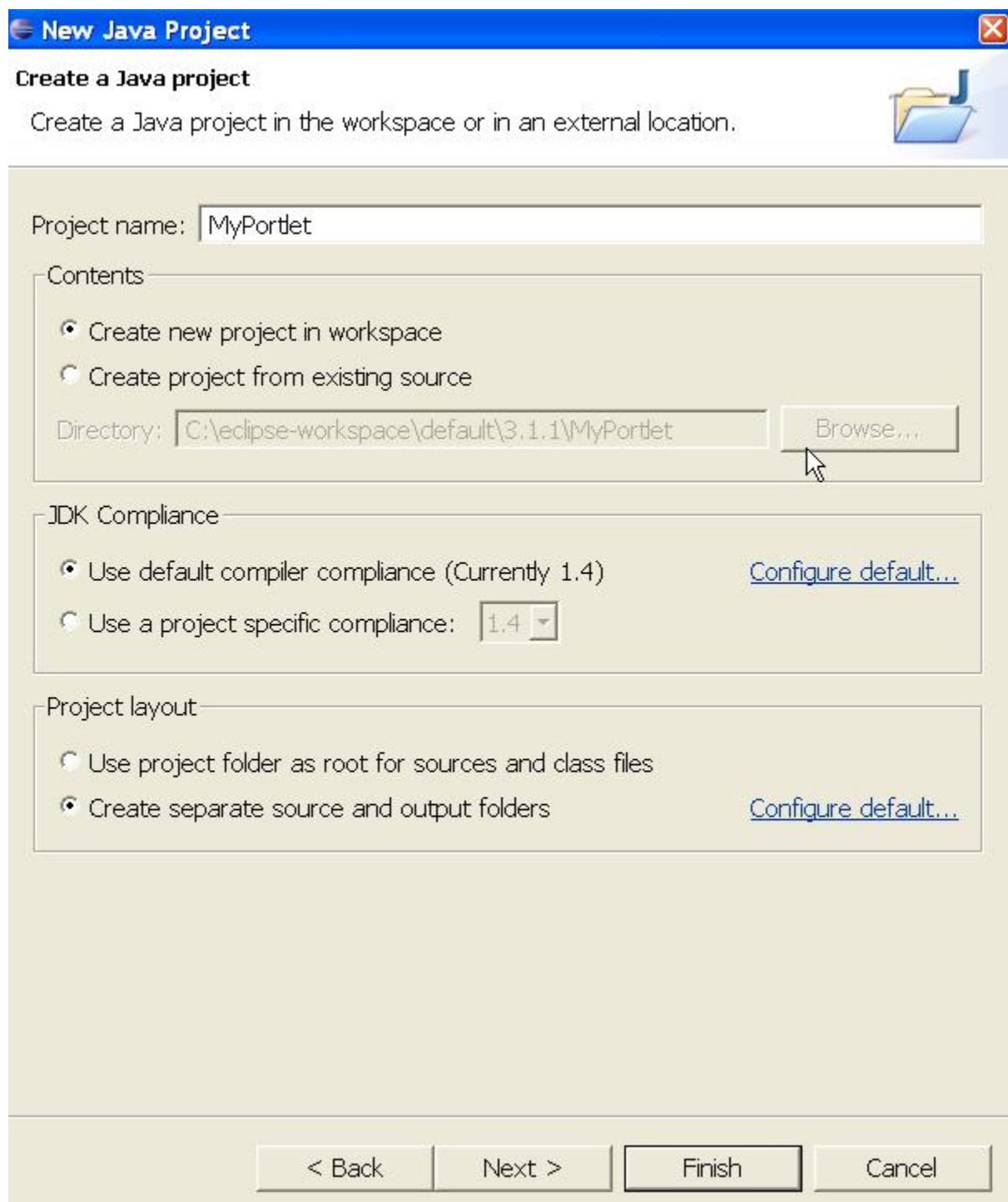
JBoss Portal 2.2 can be found at <http://www.jboss.com/products/jbossportal/downloads>.

Creating the project

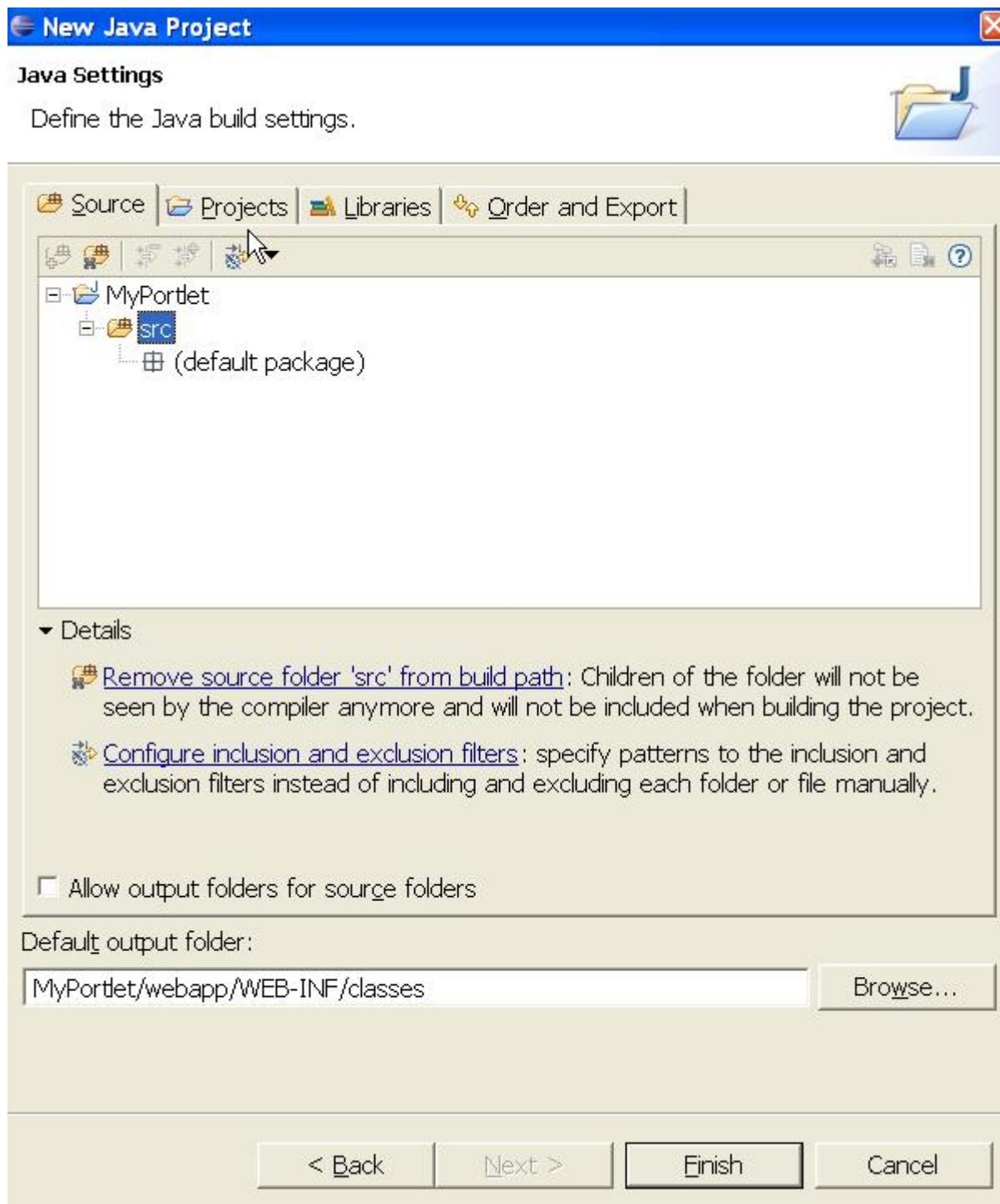
A Portlet application is basically packaged as a regular web application, but with an additional descriptor; portlet.xml. The first step of the tutorial is to create the project structure in eclipse. First, let's create the Java project itself using the new project wizard. We call the project 'MyPortlet'. Make sure to select the "Create separate source and output folders" radio button, and hit "next". In the next wizard step, set the

output folder for the 'src' source folder to 'MyPortlet/webapp/WEB-INF/classes'. This makes sure it will be easy for us to export the application as a WAR file when we're done.

New project wizard



New project wizard, cont



Classpath settings

Before building the application itself, we need to add some required jar files to the build classpath and the WEB-INF/lib folder. Firstly, create the WEB-INF/lib folder and download the WebWork 2.2.1 distribution and unzip it to your local harddrive. Locate the jar files shown in the screenshot and put them in the newly created WEB-INF/lib folder. Select all the jar files, and right click and select "Build Path -> Add to Build Path". Now your local project should look similar to the screenshot.



If there are jar files here that aren't included in the webwork distribution you have downloaded, you can safely assume that they are not needed.

portlet.xml

Next thing we do is create a portlet.xml file in the WEB-INF folder. In this file, write the following:

```
<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>
      <!-- The view mode namespace. Maps to a namespace in the xwork config file -->
      <name>viewNamespace</name>
      <value>/view</value>
    </init-param>
    <init-param>
      <!-- The default action to invoke in view mode -->
      <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>
```

This portlet.xml file sets up the portlet using the `com.opensymphony.webwork.portlet.dispatcher.Jsr168Dispatcher` Portlet implementation. It also tells the Portlet that it will map the `view` portlet mode to a `/view` namespace in the XWork configuration, which we must remember when building our XWork actions. In addition, it tells the portlet that if it does not find an action parameter in the portlet request, the default action to invoke is the "index" action, which should reside in the `/view` namespace in our xwork configuration.

web.xml

The WebWork Portlet support also requires you to include a web.xml descriptor that sets up some special servlets and filters needed to enable support for the WebWork tag libraries and template languages, since it relies on some of the interfaces and classes in the Servlet API. So create a web.xml file in the WEB-INF folder, and add the following:

```

<?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-2.2.2.jar</taglib-location>
    </taglib>
</web-app>

```

The FilterDispatcher makes sure that URLs to stylesheets and js files within the webwork jar file resolve correctly. The ServletContextHolderListener is a Servlet context listener that stores a reference to the servlet context of the web application. This is needed by some of the initialization procedures used in the WebWork Portlet. The 'preparator' servlet is a special servlet that, before dispatching to a view (like JSP/ftl or velocity) initializes the HttpServletRequest/Response, and other Servlet API classes in the ServletActionContext that is used in many of the JSPs and templates.

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:

```
<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:

```
<?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-portlet-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>
```



If you're using version 2.2.1 of WebWork, include the file webwork-default.xml instead.

Some important things to notice are that we create a package with namespace *view*, and our package extends the *webwork-portlet-default* package. The *webwork-portlet-default* package contains some special result types needed to run WebWork/XWork in a portlet container.

JBoss Portal descriptors

In addition to the regular *portlet.xml* and *web.xml* descriptors, JBoss Portal 2.2 requires us to add a couple of JBoss specific descriptor files. One of these descriptor files is named according to the name of the context root of our application, which in this case is the name of the exported war file itself. We will later create a war file named MyPortlet.war, so the name of the JBoss descriptor becomes 'MyPortlet-object.xml'. So we create this file in the WEB-INF folder, and insert the following text:

```
<?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>
</deployments>
```

```
</page>
</deployment>
<deployment>
    <if-exists>overwrite</if-exists>
    <instance>
        <instance-name>MyPortletInstance</instance-name>
        <component-ref>MyPortlet.MyPortlet</component-ref>
    </instance>
</deployment>
</deployments>
```

In addition, we need two other files, *jboss-app.xml* and *jboss-portlet.xml* which looks like this:

```
<jboss-app>
    <app-name>MyPortlet</app-name>
</jboss-app>
```

```
<portlet-app>
    <portlet>
        <portlet-name>MyPortlet</portlet-name>
        <security>
            </security>
    </portlet>
</portlet-app>
```

Deployment

Now we have a project structure that looks like this:

Project structure



Now it's time to try our incredible HelloWorld portlet. In a Windows Explorer session, we select the WEB-INF folder and zip it up and name the file 'MyPortlet.war'. Drop this war file in the server/default/deploy folder of JBoss Portal, and start the server. By default, the URL for JBoss portal is <http://localhost:8080/portal>, so point your browser to this address, and you will get to the front page of the portal, where you should get a "MyPortlet Tutorial" menu entry, as shown in the screenshot below. When pressing this menu link, you will get to our fantastic "Hello World" page!

JBoss Portal front page

The screenshot shows the JBoss Portal interface. At the top, there is a navigation bar with tabs: MyPortlet Tutorial, PortletWork Example, Test, WebWorkPortlet Example, and default. Below the navigation bar, there are several portlets:

- User portlet**: A login form with fields for Standard Login and a message: "Don't have an account yet? You can create one".
- Pages**: A list of pages: default, WebWorkPortlet Example, PortletWork Example, Test, and MyPortlet Tutorial.
- Select the Portal Theme**: A theme selection interface showing available themes: portal Industrial, portal Nphalanx, portal Maple, and portal mission-critical.
- JBoss Portal**: The main content area. It contains a brief introduction: "JBoss Portal provides an open source platform for hosting and serving a...". It also features links to Support Services (with a 3D cube icon), PortletSwap (with a globe icon), and Project Information (with a lightning bolt icon). Below this, a thank you message is displayed: "Thank you for downloading and deploying JBoss Portal. We hope you enjoy it!" followed by the names "Bacci e Abracci, The JBoss Portal Team".

At the bottom of the page, a footer bar indicates: "Powered by JBoss Portal" and "Theme by Novell".

MyPortlet portlet page

The screenshot shows the JBoss Portal interface with the MyPortlet portlet selected. The content area displays the message "Hello world!".

At the bottom of the page, a footer bar indicates: "Powered by JBoss Portal" and "Theme by Novell".

Next step

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:

```
<%@ 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>
```

Now we're ready to code some Java, not much, but at least a little bit. We create a new package in our `src` folder, let's name it `com.opensymphony.webwork.portlet.tutorial`. In this package, create a `HelloWorldAction` class. In usual WebWork manners, this class extends the `ActionSupport` class from the XWork framework, and we'll add a couple of properties that maps to our form in the JSP we just created:

```
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:

```
<%@ taglib prefix="ww" uri="/webwork" %>

<H2>Hello <ww:property value="firstName" /> <ww:property value="lastName" /></H2>
<p/>
<a href="">Back to form</a>
```

Re-deployment

Now we're ready to do a re-deployment of our application, so zip up a new war and drop it in the `server/default/deploy` folder. The "MyPortlet Tutorial" page will now display:

Hello World form



Enter some data, and press the "Say hello!" button, and you will get a nice little personalized "hello" message:

Personalized Hello World



Tiles Use

This page last changed on Feb 27, 2006 by [phil](#).

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("tilesDefinitions"));
            try {
                TilesUtil.createDefinitionsFactory(evt.getServletContext(), factoryConfig);
            } catch (DefinitionsFactoryException e) {
                e.printStackTrace();
            }
            initialized = true;
        }
    }

    public void contextDestroyed (ServletContextEvent evt) {
    }
}
```

TutorialLesson05

This page last changed on Feb 04, 2005 by [plightbo](#).

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 Dec 11, 2005 by [plightbo](#).

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 Scriptlets:

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

<code><ww:property /></code>	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

<code><ww:iterator /></code>	Iterates over a collection.
<code><ww:generator /></code>	Generates iterators.
<code><ww:append /></code>	Appends several iterators.
<code><ww:subset /></code>	Gets a subset of an iterator.
<code><ww:merge /></code>	Merges several iterators into one.
<code><ww:sort /></code>	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

<code><ww:audio /></code>	???
<code><ww:prompt /></code>	???
<code><ww:filled /></code>	???
<code><ww:log /></code>	???

Volunteers needed to write this part.

Internationalisation tags

<code><ww:text /></code>	Prints out an internationalized string.
<code><ww:i18n /></code>	Places a resource bundle on the Value Stack, for access by the text tag.

[Previous Lesson](#) | [Next Lesson](#)

TutorialLesson04-01-01

This page last changed on Jan 07, 2006 by [plightbo](#).

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:form>


```

```

<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 to 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 />	<i>Help needed here.</i>
<ww:token />	<i>Help needed here.</i>

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($parameters.size)" #end
        #if ($parametersmaxlength) maxlength="$!webwork.htmlEncode($parametersmaxlength)" #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](#)

TutorialLesson04-02

This page last changed on Dec 13, 2005 by [sstephens](#).

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 without 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](#)

TutorialLesson04-03

This page last changed on Feb 04, 2005 by [richardbondi](#).

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="${wwUtil.buildUrl('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 Ffreemarker 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 Feb 04, 2005 by [plightbo](#).

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 scratch:

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.WebWorkConversionErrorInterceptor"/>
            <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-stack>
        </interceptors>
    </package>
</xwork>
```

```

        <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 Mar 22, 2006 by [mrdon](#).

Actions

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 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. 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.

Understanding interceptors

This page last changed on Mar 22, 2006 by [mrdon](#).



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,

!overview.png!



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.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 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 IoC

This page last changed on Dec 21, 2006 by [phil](#).

What is IoC or DI ?

IoC stands for Inversion of Control, or DI, Dependency Injection. The term was founded by [Martin Fowler](#), and is now currently supported by several IoC containers, such as Pico, XWork and Spring. IoC allows us to reduce the coupling between components and classes, makes unit testing a whole lot easier and generally promotes good coding style.

But what does it do ? Like the name suggests, it will inject certain dependant objects, rather than requiring the depending object to retrieve those objects itself.

An example

Suppose you need a JDBC connection (or Hibernate Session, or ..) in your WebWork Action to communicate with a database. Normally, you would be required to set up a Driver Manager, register a JDBC Driver and get a connection from it.



This is pseudo code, for demonstration only.

```
..  
if (driverManager == null){  
    driverManager = new DriverManager();  
    driverManager.registerDriver("jdbc.Driver");  
}  
. . .  
if (connection == null){  
    ..  
    connection = driverManager.getConnection("jdbc://host/db", username, password);  
}  
..
```

Anyway, you get the point. That's a lot of work to get that done. Now, you might want to use some static helper class, store the connection in a ThreadLocal or Session, .. wouldn't it be nice if we could turn this whole 'get the connection' into 'just give me a connection' ?
Confused again ?

Let's say we develop an Interface that will contain a setter for a JDBC connection:

```
public interface JDBCConnectionAware {  
    public void setJDBCConnection( JDBCConnection connection );  
}
```

Now, if we have a class that implements this interface, we have a binding contract that allows us to set a JDBCConnection object on that depending object.

```
public class MyAction implements JDBCConnectionAware {
```

```

public JDBCConnection connection;
.
.
public String execute() {
    .
    .connection.prepareStatement("..."); //here we use the connection object
    .
    return Action.SUCCESS;
}

//as required by the implemented JDBCConnectionAware interface
public void setJDBCConnection(JDBCConnection connection){
    this.connection = connection;
}
}

```

So, if look at this example, you see that we do not have to get the connection object from somewhere, nor set it up. We just assume that someone will 'give' us a connection, or 'inject' it. That 'someone' will be our IoC container.

The IoC container will take care of instantiating objects and singletons, and when it finds a object to be implementing a certain interface, it will do the injection of the dependency.



This example is meant to demonstrate the injection. The loose coupling and easy testing is explained a whole lot better by [Martin Fowler's examples](#).

Example in XWork

Coming soon.

Example in Spring

Coming soon.

Example in Pico

Coming soon.

Understanding results

This page last changed on Mar 22, 2006 by [mrdon](#).

A result is a piece of code that is executed after your action has already completed and returned a value such as success or error. But WebWork comes with most of the result you'll need, for example: such as "servlet dispatcher," used for JSPs, and Velocity as well as alternative results such as FreeMarker and Jasper Reports (in PDF, XML, and 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>
```

As you can see, that result configuration is made up of two parts: result mappings, which you've already seen associated with an action mapping, and result types.

Configuring result types

Every package in WebWork can be associated with one or more result types.

```
<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>
```

Reducing configuration duplication with global result mappings

Another way to reduce the amount of configuration in xwork.xml is through the use of global result mappings. Web applications often have a common set of results that are used across many actions. Common results include redirects to login actions and permission-denied pages. Rather than define each of these results in every action mapping, WebWork lets you centralize the definitions for the common pages.

```
<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>
```



Be Careful

Because global results are searched after local results, you can override any global result mapping by creating a local result mapping for a specific action. Recall that results can point to locations using relative or absolute paths. Because you may not know the context in which they're being invoked, it's best to use absolute paths for global results.

Understanding tag libraries

This page last changed on Dec 21, 2006 by [phil](#).

What are tag libraries ?

Tag libraries are collections of special tags that add extra functionality to your view layer (mostly JSP's). These special tags are proceeded by a namespace (ww:, webwork:, jsp:, ..) and are provided to reduce the code you have to write in your view layer.

WebWork Tag library

WebWork tags are no different than any other tag library. You declare them at the top of your page:

```
<%@ taglib prefix="ww" uri="/webwork" %>
```

And then you can use them to display (bean) properties, iterate collections, create forms, show I10n text, create dropdown boxes, and a whole lot more.

WebWork tags come in different flavors. Some are what we call form tags; they can be used to create complex forms very quickly and in only a few lines that would normally require a great deal of complex code. Other tags are used to control the flow (if/else), control data (bean, push, set) or just display it (property). WebWork contains a lot of tags, and good knowledge of them will give your productivity a huge boost.

About themes

Form tags use so-called themes: these themes add extra markup to your tags. One theme might just add tables and labels, while another theme might allow ajax functionality. The default theme is xhtml, unless you override it in the theme attribute, the parent tag's attribute or the webwork.properties file. The xhtml theme will add labels, validation errors and uses a table-based layout.

For example:

```
<ww:textfield name="test" value="%{test}" label="getText('name')"/>
```

In simple mode:

```
<input type="text" name="test" value="test" id="foo_test"/>
```

In xhtml mode:

```
<tr>
```

```

<td class="tdLabel">
    <label for="foo_test" class="label">Name:</label>
</td>
<td>
    <input type="text" name="test" value="test" id="foo_test"/>
</td>
</tr>

```

This clearly shows that a good theme will greatly reduce the amount of code you have to write. Therefore, if you find yourself in need of a special layout (say, a three column layout), you'll often find it a lot easier in the long run to create a custom theme (or override one), than using scriptlets or using the simple theme.

Gotcha's

There are some things you'll need to keep in mind while using the ww tags. Some might not entirely work as you expect at first.

Take the `ww:textfield` tag again (see above). You'll notice that the `value` attribute has a value of `%{test}`. The `%{..}` tells WW to look up the property on the `valuestack` (resulting in a call to the `getTest()` method on the Action) rather than taking the String literal 'test' (notice the difference with the `name` attribute, which does in fact use the String 'test'. This is all pretty normal.

However, there are some places where the `%{..}` notation is not required; take the `ww:iterator` tag. It takes a `value` attribute, so you could use it to iterate a collection you get from the `valuestack`. For example, `getNames()` returns a `List` with names. Now, since there is no possibility to iterate a String literal, you can drop the `%{..}` notation, and just use `value="names"` instead of `value="%{names}"` (although the latter will work as well). Still no biggie.

However, there are some tags where things will work differently from what you expect. The `ww:set` tag, for example, allows you to store a certain object in a different scope. Let's say we want to store a message in our session. This is what you might expect to work:

```
<ww:set name="message" value="all your base are belong to us" scope="session"/>
```

Unfortunatealy, this will not work. Nothing will get stored in the session. Why ?

The reason for this is that the `value` attribute expects an expression rather than a String literal. Therefor, if you want to store this message, you'll have to ask the `valuestack` to create a String for you:

```
<ww:set name="message" value="%{'all your base are belong to us'}" scope="session"/>
```

Therefore, it's very important to read the attribute information for each tag (look them up in the documentation!). For example, for the `ww:set` tag, this is:

- `name`: String
- `value`: Object
- .. (others skipped for brevity)



Keep this in mind whenever a tag does not function like you would expect, esp. when dealing with booleans !

Tutorial

Coming soon.

Understanding validation

This page last changed on Jan 19, 2006 by [mszklano](#).

In absence of product documentation in this area, you can have a look at excellent article entitled [Webwork Validation](#) on [java.net](#).

WebWork Continuous Integrations

This page last changed on Jan 28, 2007 by [tm_jee](#).

WebWork is building on following Continuous Integration Build System.

Viewtier Parabuild

- [WebWork CVS Head](#)

Special thanks to Slava (Viewtier) for making this possible.

Atlassian Bamboo

- [WebWork CVS Head](#)
- [XWork Head](#)
- [XWork 1-2 Branch Head](#)

Special thanks to Mike & Edwin (Atlassian) for making this possible.