Lightweight J2EE Framework

Struts, spring, hibernate

Software System Design
Zhu Hongjun

Session 3: Struts MVC

- OGNL and Value Stack
- Struts Tags
- Type Conversion
- Message Handling and I18n

OGNL and Value Stack

- OGNL
  - Object Graph Navigation Language, an expression language for getting and setting properties of Java objects
  - OGNL started out as a way to set up associations between UI components and controllers using property names

OGNL and Value Stack

- OGNL (cont.)
  - Several of the uses to which OGNL has been applied are
    - A binding language between GUI elements (textfield, combobox, etc.) to model objects
    - A data source language to map between table columns and a Swing TableModel
    - Transformations are made easier by OGNL's TypeConverter mechanism to convert values from one type to another

OGNL and Value Stack

- OGNL (cont.)
  - The framework uses a standard naming context to evaluate OGNL expressions
  - OGNL has a notion of there being a root (or default) object within the context
  - In expression, the properties of the root object can be referenced without any special "marker" notion. References to other objects are marked with a pound sign (#)

OGNL and Value Stack

- OGNL (cont.)
  - Accessing Static Properties
    - OGNL supports accessing static properties as well as static methods
    - By default, Struts 2 is configured to disallow this--to enable OGNL's static member support you must set the struts.ognl.allowStaticMethodAccess constant to true via any of the Constant Configuration methods

OGNL and Value Stack

- OGNL (cont.)
  - Accessing Static Properties (cont.)
    - Accessing static properties
      - @some.package.ClassName@FOO_PROPERTY
      - Or @vs@FOO_PROPERTY
    - Accessing static method
      - @some.package.ClassName@someMethod()
      - Or @vs@someMethod()

Accessing Static Properties Demo by OGNL

```
<constant name="struts.ognl.allowStaticMethodAccess" value="true">
</constant>
```

```
<s:property value="@water.action.LoginAction@testValue" />
<br>
<s:property value="@java.lang.Math@PI" />
<br>
<s:property value="@water.action.LoginAction@getHelloValue()" />
```

```
public static String testValue = "TestValue";

public static String getHelloValue() {
    return "Hello Static";
}
```

TestValue
3.141592653589793
Hello Static

OGNL and Value Stack

- OGNL (cont.)
  - Dealing with Collections
    - Dealing with Collections (Maps, Lists, and Sets) in the framework comes often
    - To determine if an element exists in a Collection, use the operations in and not in
    - To select a subset of a collection (called projection), use a wildcard within the collection
      - ?, ^, $

Dealing with Collections by OGNL

```
person.relatives.{? #this.gender == 'male'}
```

```
<s:select label="label" name="name" list="{'name1','name2','name3'}" value="%{'name2'}" />
```

```
<s:select label="label" name="name" list="#{'foo':'foovalue', 'bar':'barvalue'}" />
```

```
<s:if test="'foo' in {'foo','bar'}">
   muhahaha
</s:if>
<s:else>
   boo
</s:else>

<s:if test="'foo' not in {'foo','bar'}">
   muhahaha
</s:if>
<s:else>
   boo
</s:else>
```

List, Map, Sets

OGNL and Value Stack

- OGNL (cont.)
  - Lambda Expressions
    - OGNL supports basic lamba expression syntax enabling you to write simple functions

```
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
```

```
<s:property value="#fib =:[#this==0 ? 0 : #this==1 ? 1 : #fib(#this-2)+#fib(#this-1)], #fib(11)" />
```

OGNL and Value Stack

- Value Stack
  - Struts sets the OGNL context to be our ActionContext, and the value stack to be the OGNL root object
  - Along with the value stack, the framework places other objects in the ActionContext, including Maps representing the application, session, and request contexts
  - These objects coexist in the ActionContext

Objects in Context Map Demo

```
                |
                |--application
                |
                |--session
context map---  |
                |--value stack(root)
                |
                |--action (the current action)
                |
                |--request
                |
                |--parameters
                |
                |--attr (searches page, request, session, then application scopes)
                |
```

Value Stack and Stack Context Demo

[Debug]

Struts ValueStack Debug

Value Stack Contents

Object	Property Name	Property Value
water.action.LoginAction	loginPassword	ces
	loginName	abc
com.opensymphony.xwork2.DefaultTextProvider	texts	null

Stack Context

These items are available using the #key notation

Key	Value
com.opensymphony.xwork2.dispatcher.HttpServletRequest	org.apache.struts2.dispatcher.StrutsRequestWrapper@12997e7
com.opensymphony.xwork2.ActionContext.locale	zh_CN
com.opensymphony.xwork2.dispatcher.HttpServletResponse	org.apache.catalina.connector.ResponseFacade@13281c9
com.opensymphony.xwork2.ActionContext.name	login
	{org.apache.tomcat.JarScanner=org.apache.tomcat.util.scan.StandardJarScanne freemarker.Configuration=freemarker.template.Configuration@12abfb4, javax.servlet.context.tempdir=D:\J2EEWorkspace\.metadata\.plugins\org.eclipse

OGNL and Value Stack

- Value Stack (cont.)
  - The Action instance is always pushed onto the value stack
    - References to Action properties can omit the # marker
    - Other (non-root) objects in the ActionContext can be rendered use the # notation
  - The ActionContext is also exposed to Action classes via a static method

Accessing root object and non-root objects Demo

```
<s:textfield name="loginName" label="Login Name"></s:textfield>
<s:textfield name="loginPassword" label="Login Password"></s:textfield>
```

```
ActionContext.getContext().getSession()
        .put("loginName", getLoginName());
```

```
public String getLoginPassword() {
    return loginPassword;
}

public void setLoginPassword(String loginPassword) {
    this.loginPassword = loginPassword;
}

private String loginPassword;
```

```
<s:property value="#session.loginName" />
<s:property value="loginPassword" />
```

Struts Tags

- The framework provides a tag library decoupled from the view technology
- 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
- The types of tags can be broken in to two types: generic and UI

Struts Tags

- UI Tags
  - UI tags do not provide much control structure or logic. they display data in rich and reusable HTML
  - All UI tags are driven by *templates and themes*
  - Template support allows UI tags to build a rich set of reusable HTML components that can be customized to fit exact requirements

Struts Tags

- UI Tags (cont.)
  - Themes and Templates
    - Tag
      - A small piece of code executed from whithin JSP, FreeMarker, or Velocity
    - Templates
      - A bit of code, usually written in FreeMarker, that can be rendered by certain tags (HTML tags)
    - Themes
      - A collection of *templates* packaged together to provide common functionality

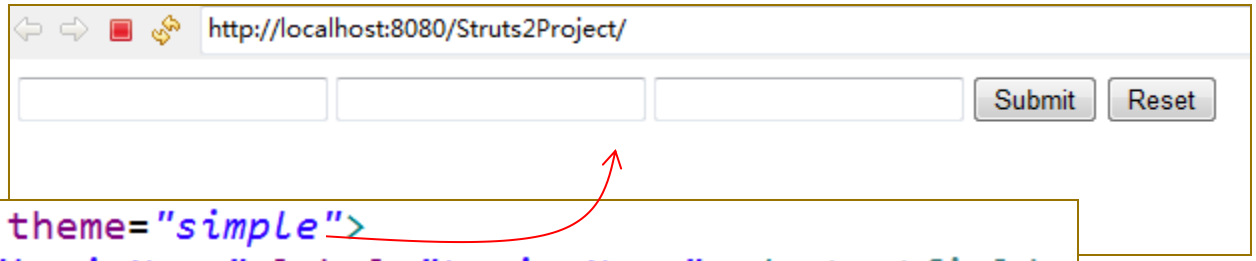
Struts Tags

- UI Tags (cont.)
  - Themes and Templates (cont.)
    - The template directory is defined by the struts.ui.templateDir property in struts.properties (defaults to template)
    - The struts.ui.theme property in struts.properties defaults to *xhtml*
      - Simple theme
      - Xhtml theme
      - Css xhtml theme
      - Ajax theme

Struts Tags

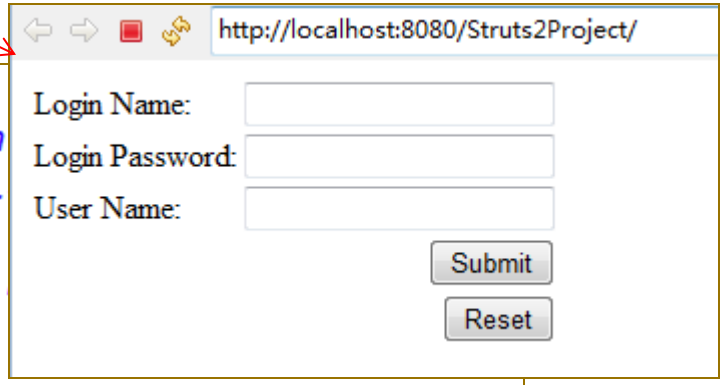
- UI Tags (cont.)
  - Form Tags
    - Within the form tags, there are two classes of tags: the form tag itself, and all other tags, which make up the individual form elements
    - Form Tag Themes
      - Simple
      - Xhtml
      - Ajax

Form Theme Demo




```
<s:form action="login" theme="simple">
    <s:textfield name="loginName" label="Login Name"></s:textfield>
    <s:textfield name="loginPassword" label="Login Password">
    </s:textfield>
    <s:textfield name="user.name" label="User Name"></s:textfield>
    <s:submit label="Login"></s:submit>
    <s:reset label="Reset"></s:reset>
</s:form>
```

```
<s:form action="login" theme="xhtml">
    <s:textfield name="loginName" label="Login
    <s:textfield name="loginPassword" label="L
    </s:textfield>
    <s:textfield name="user.name" label="User
    <s:submit label="Login"></s:submit>
    <s:reset label="Reset"></s:reset>
</s:form>
```



Struts Tags

- UI Tags (cont.)
  - Form Tags (cont.)
    - Common Attributes
      - Template related attributes
      - Javascript related attributes
      - Tooltip related attributes
      - General attributes
    - Some tag attributes may not be utilized by all, or any, of the templates

Common Attributes

General Attributes

Attribute	Theme	Data Types	Description
cssClass	simple	String	define html class attribute
cssStyle	simple	String	define html style attribute
cssClass	simple	String	error class attribute
cssStyle	simple	String	error 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
requiredposition	xhtml	String	define required label position of form
name	simple	String	Form Element's field name mapping
required	xhtml	Boolean	add * to label (true to add false othe
tabIndex	simple	String	define html tabindex attribute
value	simple	Object	define value of form element

Tooltip Related Attributes

Attribute	Data Type	Default	Description
tooltip	String	none	Set the tooltip of
jsTooltipEnabled	String	false	Enable js tooltip
tooltipIcon	String	/struts/static/tooltip/tooltip.gif	The url to the too
tooltipDelay	String	500	Tooltip shows up after the specified timeout (miliseconds). A behavior similar to that of OS based tooltips.
key	simple	String	The name of the property this input field represents. This will auto populate the name, label, and value

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
ondblclick	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

Struts Tags

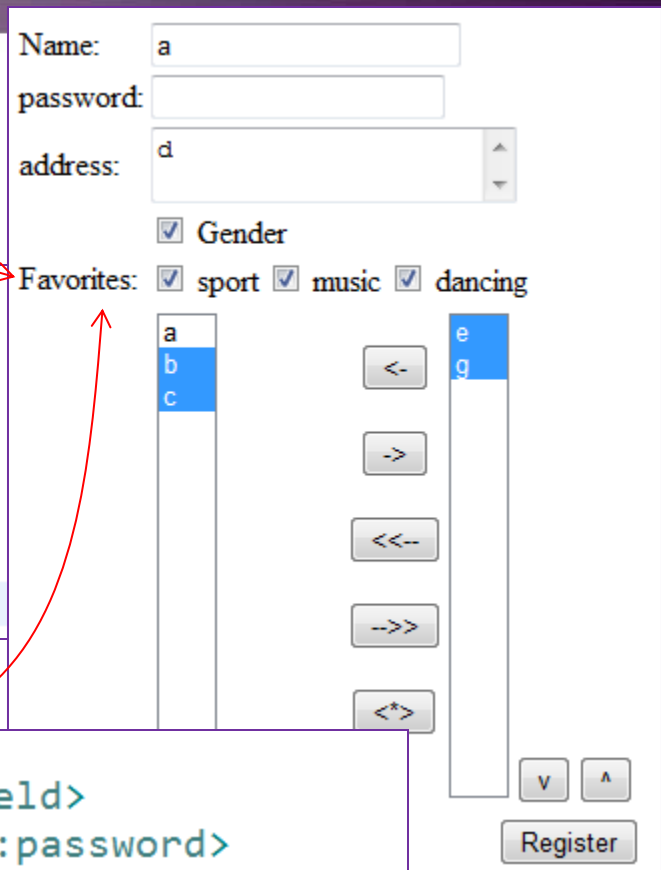
- UI Tags (cont.)
  - Form Tags
    - textfield, textarea, password, label
    - checkbox, checkboxlist, combobox, radio, select
    - doubleselect, inputtransferselect, optiontransferselect, optgroup, updownselect
    - head, hidden, reset, submit, token, file
  - Non-form UI Tags
    - actionerror, actionmessage, component, div, fielderror

Form UI tags Demo

```java
private String name, password, address;
private boolean male;
private String[] favorites = new String[] { "sport", "music",
        selectedFavorites;
private String[] leftList = new String[] { "a", "b", "c" },
        rightList = new String[] { "e", "g" },
        leftResult,
        rightResult;
```

```
<s:form action="welcome">
    <s:textfield name="name" label="Name"></s:textfield>
    <s:password name="password" label="password"></s:password>
    <s:textarea name="address" label="address"></s:textarea>
    <s:checkbox name="male" label="Gender"></s:checkbox>
    <s:checkboxlist list="favorites" name="selectedFavorites"
        label="Favorites" value="selectedFavorites"></s:checkboxlist>

    <s:optiontransferselect name="leftResult" doubleList="rightList"
        list="leftList" doubleName="rightResult" value="leftResult"
        doubleValue="rightResult"></s:optiontransferselect>
    <s:submit value="Register"></s:submit>
</s:form>
```



Struts Tags

- UI Tags (cont.)
  - Ajax tags
    - a
    - autocompeleter
    - bind
    - datetimepicker
    - div
    - head
    - submit/tabbedpanel/textarea/tree/treenode

Ajax Tags Demo

```
<%@ taglib prefix="s" uri="/struts-tags"%>
<%@ taglib prefix="sx" uri="/struts-dojo-tags"%>
```

```
<head>
<meta http-equiv="Content-Ty
<title>Insert title here</ti
<sx:head />
</head>
```

```
<s:form action="login" theme="ajax" id="form">
    <s:textfield name="loginName" label="Login Name"></s:textfield>
    <s:password name="loginPassword" label="Login Password">
    </s:password>
</s:form>
<sx:submit formId="form"></sx:submit>
```

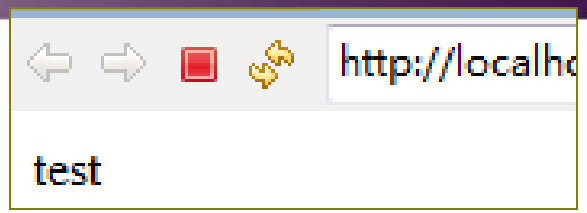
Struts Tags

- Generic Tags
  - Generic tags are used for controlling the execution flow when the pages render
  - These tags also allow for data extraction from places other than your action or the value stack, such as Localization, JavaBeans, and including additional URLs or action executions

Struts Tags

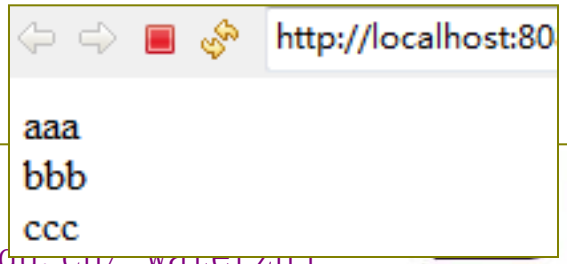
- Generic Tags
  - Control tags
    - If elseif else
    - Append, Generator, Iterator
    - Merge, Sort, subset
  - Data tags
    - A, action, bean, date, debug, i18n
    - Include, param, property, push, set, text, url

Generic Tags Demo




```
<s:set name="result" value="%{'test'}"></s:set>
<s:if test="%{#result=='test'}">
    <s:property value="#result" />
</s:if>
<s:else>
    <s:property value="%{'cde'}" />
</s:else>
```

```
<s:generator val="%{'aaa,bbb,ccc,ddd,eee'}" separator="," count="3">
    <s:iterator>
        <s:property />
        <br />
    </s:iterator>
</s:generator>
```



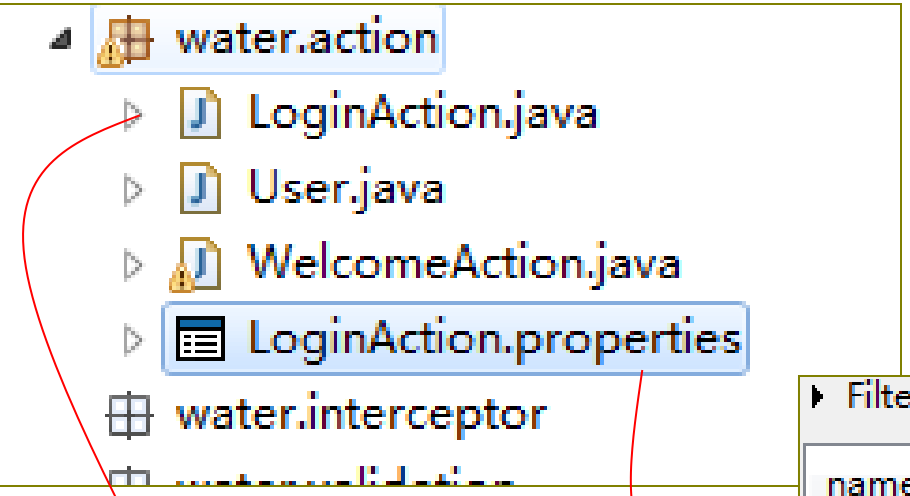
Type Conversion

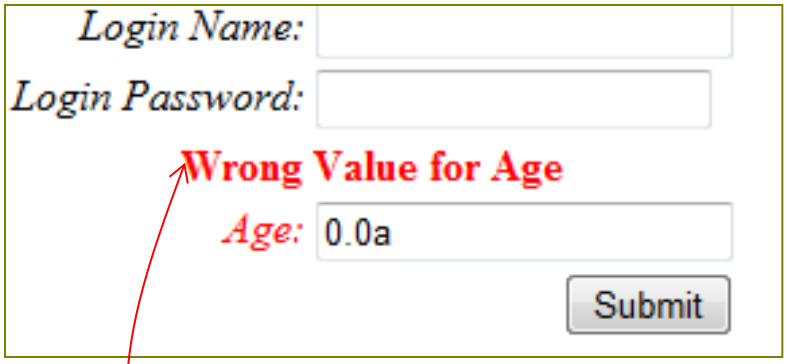
- Routine type conversion in the framework is transparent
- Generally, all you need to do is ensure that HTML inputs have names that can be used in OGNL expressions
- XWork will automatically handle the most common type conversion for you

Type Conversion

- Built in Type Conversion Support
  - Includes support for converting to and from Strings for each of the following
    - String, boolean, char, int, double, float, short, long
    - Dates, arrays, collections
  - Customing Conversion Error Message
    - 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

Customing Type Conversion Error Message Demo





Filter

name	value
invalid.fieldvalue.age	Wrong Value for Age

Action class must extend ActionSupport for capturing type conversion error message

Type Conversion

- Creating a Type Converter
  - Two steps
    - Create a type converter by extending StrutsTypeConverter
    - Apply the converter to an action
      - Create a file called 'ActionClassName-conversion.properties' in the same location of the classpath as the Action class itself resides
      - Within the conversion file, name the action's property and the Converter to apply to it

Creating a Type Converter Demo

Login Name:
Login Password:
Line: 5,6,7,8
Age: 0.0
Submit

Login Name:
Login Password:
the format of line is not correct!
Line: 5,6,7
Age: 0.0
Submit

```
<s:textfield name="sl" label="Line"></s:textfield>
<s:textfield name="age" label="Age"></s:textfield>
```

```
private double age;
private StraitLine sl;

public StraitLine getSl() {
    return sl;
}

public void setSl(StraitLine sl) {
    this.sl = sl;
}

public double getAge() {
    return age;
}
```

- water
  - action
    - LoginAction-conversion.properties
    - LoginAction.java
    - LoginAction.properties
    - StraitLine.java
    - User.java

```
# syntax: <propertyName> = <converterClassName>
age=water.converter.IntegerConverter
sl=water.converter.StraitLineConverter
```

Type Conversion

- Creating a Type Converter (cont.)
  - You can also apply a type converter to a bean or model
    - When getting or setting the property of a bean, the framework will look for "classname-conversion.properties" in the same location of the **classpath** as the target bean.
  - And for an application
    - Application-wide converters can be specified in a file called xwork-conversion.properties located in the root of the classpath

Message Handling and Internationalization

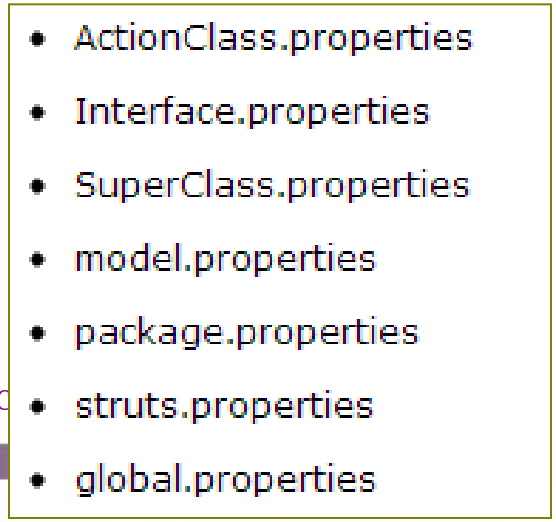
- The framework supports internationalization (i18n) in the following places
  - The UI Tags
  - Messages and Errors from the ValidationAware interface
  - Within action classes that extend ActionSupport through the getText() method

Message Handling and Internationalization

- Locales
  - ISO 3166 country codes
    - CHINA——CN
    - UNITED STATES——US
  - ISO 639 language codes
    - Chinese (Simplified)——zh
    - English——en
  - Struts2 uses resource bundles to provide multiple language and locale options

Message Handling and Internationalization

- Resource bundles are the file that contains the key/value pairs for the default language of your application
  - Naming format for a resource file
    - bundlename_languageCode_countryCode.properties
- Resource bundles search order



Message Handling and Internationalization

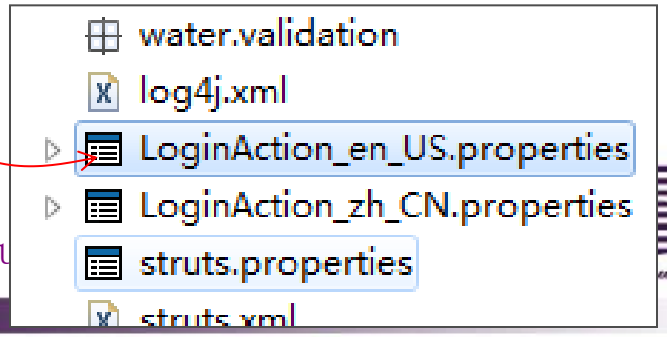
- Ways to access the message resources
  - Using getText from a tag
    - Like
```
<s:property value="getText('login_name')" />
```
  - Using text tag
    - Like
```
<s:text name="login_password">
    default message!
</s:text>
```

name	value
login_name	登录名
login_password	登录密码

Message Handling and Internationalization

- Ways to access the message resources (cont.)
  - Using i18n tag
    - The i18n tag pushes an arbitrary resource bundle on to the value stack.
    - Other tags within the scope of the i18n tag can display messages from that resource bundle

```
<s:i18n name="LoginAction_en_US">
    <s:text name="login_password">
    default message!
    </s:text>
</s:i18n>
```



Message Handling and Internationalization

- Ways to access the message resources (cont.)
  - Using the key attribute of UI tags
    - The key attribute of most UI tags can be used to retrieve a message from a resource bundle

```
<s:textfield name="user.name" key="login_name"></s:textfield>
<s:password name="loginPassword" key="login_password">
```

name	value
login_name	登录名
login_password	登录密码

Message Handling and Internationalization

- Ways to access the message resources (cont.)
  - Using getText() from validation message

```
<field name="loginPassword">
    <field-validator type="requiredstring">
        <message>${getText("login_password")} is required!</message>
    </field-validator>
</field>
```

Message Handling and Internationalization

- Ways to access the message resources (cont.)
  - Using getText() from a action class

```
public class LoginAction extends ActionSupport {

    public String login() {

        System.out.println(getText("login_name"));

        return "index";

    }
}
```

Conclusions

- OGNL and the value stack
- Struts Tags
- Type Conversion
- Message Handling and I18n