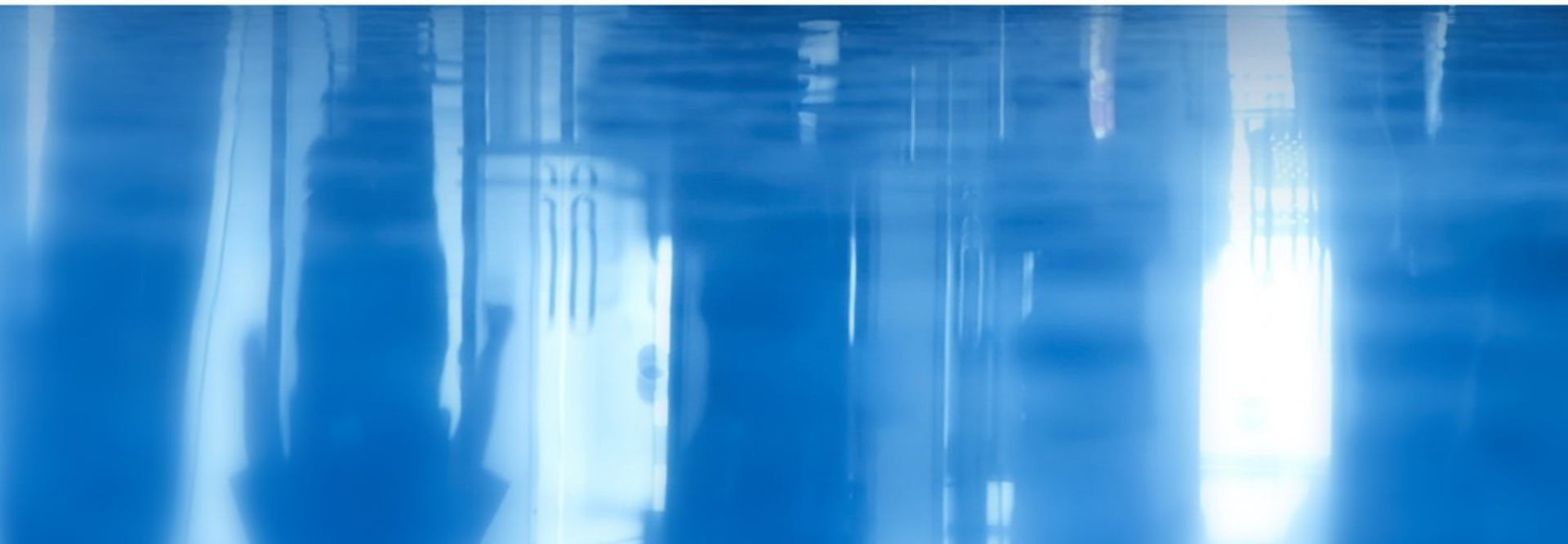


第12章

Struts 2 框架基础



本章内容

- Struts 2是基于MVC设计模式的Web应用程序开发框架，它是由Struts和WebWork发展而来的。
- 本章首先讨论Struts 2框架的体系结构，Action类的使用、OGNL表达式语言、Struts 2标签，接下来介绍Struts 2的国际化 and 用户输入校验，最后介绍了使用Tiles插件构建页面布局。

本章内容

- 12.1 Struts 2框架概述
- 12.2 注册/登录系统
- 12.3 OGNL
- 12.4 Struts 2常用标签
- 12.5 Struts 2的国际化
- 12.6 用户输入校验
- 12.7 用Tiles实现页面布局

12.1 Struts 2框架概述

- Apache Struts是用于开发Java Web应用程序的开源框架。
- Struts提供了Web应用开发的优秀框架，是世界上应用最广泛的MVC框架。然而，随着Web应用开发需求的日益增长，Struts 已不能满足需要，修改Struts框架成为必要。
- Apache Struts小组和另一个Java EE框架WebWork联手共同开发一个更高级的框架Struts 2。

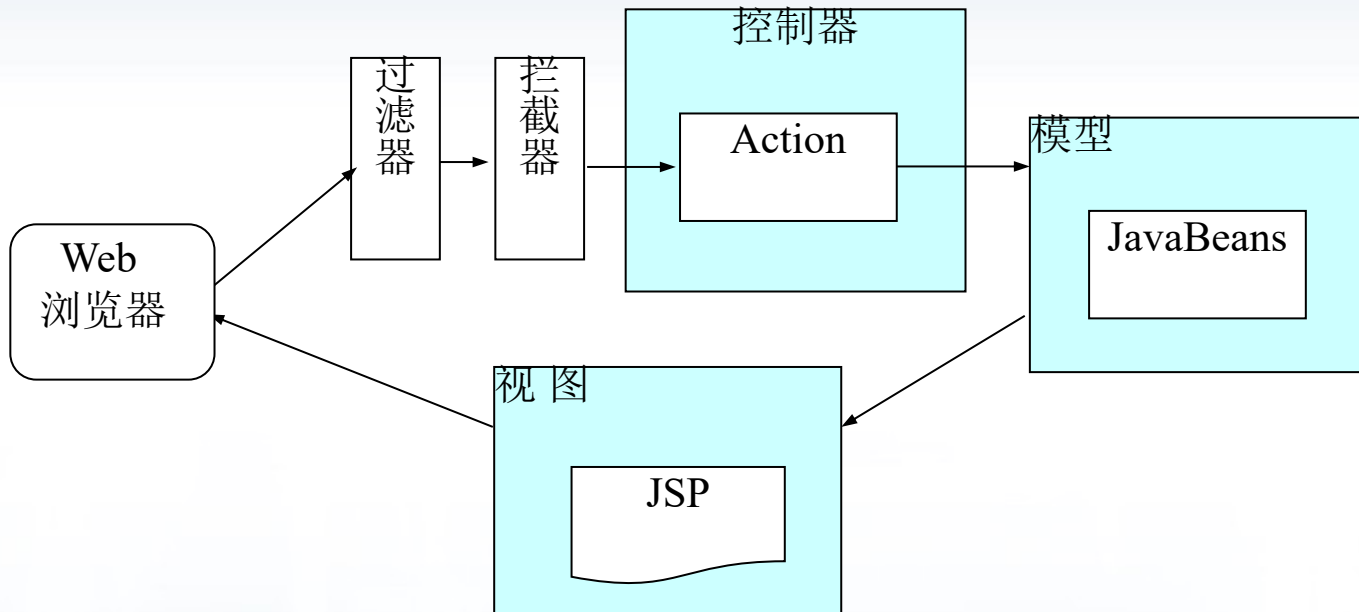
12.1 Struts 2框架概述

- Struts 2结合了Struts和WebWork的共同优点，对开发者更友好，具有支持Ajax、快速开发和可扩展等特性。它已成为构建、部署和维护动态的、可扩展的Web应用框架。Struts 2并不是Struts的简单升级，可以说Struts 2是一个既新又不新的MVC框架。
- Struts 2的设计思想和核心架构与WebWork是完全一致的，同时它又吸收了Struts的一些优点。也就是说，Struts 2是集WebWork和Struts两者设计思想之优点而设计出来的新一代MVC框架。

12.1.1 Struts 2框架的组成

- Struts 2框架是基于MVC设计模式的Web应用开发框架，它主要包括控制器、Action对象、视图JSP页面和配置文件等，如图12.1所示。

图12.1 Struts 2的MVC架构



- 视图器提供JSP页面模板，拦截器提供强大的业务逻辑，拦截器提供使用量来拦截，拦截器提供JSP页面的开发。

12.1.2 Struts 2开发环境的构建

- 运行Struts 2的平台必须满足下面需求：
Servlet API 2.4、JSP API 2.0、Java 5。

1. 下载Struts 2库文件

- 开发Struts 2应用程序必须安装Struts 2库文件。可以到Apache Struts Web站点下载库文件包，地址为：
<http://struts.apache.org/downloads.html>。
- 目前的最新版本是2.3.8。

1. 下载Struts 2库文件

- 假设这里下载的是struts-2.3.8-all.zip，它是完整发布软件包，其中包括示例应用程序、文档、所有的库文件和源代码。
- 将该文件解压到一个临时目录中，其中lib目录中存放的是Struts 2的所有库文件，将下列文件复制到WEB-INF\lib目录中：

1. 下载Struts 2库文件

- asm-3.3.jar
- asm-commons-3.3.jar
- asm-tree-3.3.jar
- commons-fileupload-1.2.2.jar
- commons-io-2.0.1.jar
- commons-lang3-3.1.jar

1. 下载Struts 2库文件

- freemarker-2.3.19.jar
- javassist-3.11.0.GA.jar
- ognl-3.0.6.jar
- struts2-core-2.3.8.jar
- xwork-core-2.3.8.jar

2. 在web.xml中添加过滤器

- 要使Web应用程序支持Struts 2功能，需要在web.xml文件中声明一个核心过滤器类和映射，代码如下：

2. 在web.xml中添加过滤器

```
<filter>
```

```
  <filter-name>struts2</filter-name>
```

```
  <filter-class>
```

```
    org.apache.struts2.dispatcher.ng.filter.StrutsPr  
    epareAndExecuteFilter
```

```
  </filter-class>
```

```
</filter>
```

```
<filter-mapping>
```

```
  <filter-name>struts2</filter-name>
```

```
  <url-pattern>/*</url-pattern>
```

```
</filter-mapping>
```

3. 创建struts.xml配置文件

- 配置文件用来指定URL、Java类和视图页面（如index.jsp）之间的关系。在开发环境下配置文件应保存在src目录中。
- 下面是struts.xml文件代码：

3. 创建struts.xml配置文件

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration
    2.0//EN"
    "http://struts.apache.org/dtds/struts-2.0.dtd">
<struts>
    <constant name="struts.devMode" value="true" />
    <package name="basicstruts2" extends="struts-default">
        <action name="index">
            <result>/index.jsp</result>
        </action>
    </package>
</struts>
```


3. 创建struts.xml配置文件

- 提示：可以将Struts 2自带的一个名为struts2-blank应用程序导入到Eclipse中，在该应用程序中已经完成了各种配置，在此基础上进行开发会更方便。

12.1.3 Struts 2应用的开发步骤

- 开发Struts 2应用程序大致需3个基本步骤。
 - 创建Action动作类；
 - 创建结果视图；
 - 修改配置文件struts.xml。

1. 创建Action动作类

- 动作类在Struts 2中充当控制器，由它处理用户的动作。
- 当用户在视图页面中触发一个动作时（点击超链接或提交表单），请求将经由过滤器发送到一个Action动作类。
- Struts将根据配置文件struts.xml中的信息确定要执行的Action对象。

1. 创建Action动作类

- Action对象调用execute()方法执行业务逻辑或数据访问逻辑。
- Action类执行后根据结果选择一个资源发送给客户。资源可以是服务器视图页面，也可能是PDF文件、Excel电子表格等。
- 当客户请求URL与某个动作名匹配时，Struts 2将使用struts.xml文件中的映射处理请求。
- 动作映射在struts.xml文件中使用<action>标签定义。

2. 创建视图页面

- 视图用来响应用户请求、输出处理结果。视图通常使用JSP页面实现。

3. 修改struts.xml配置文件

- struts.xml文件主要用来建立动作Action类与视图的映射。在其中为每个动作定义一个映射，它根据动作名确定执行哪个Action类，根据Action类的执行结果确定请求转发到哪个视图页面。

12.1.4 一个简单的应用程序

- 假设创建一个向客户发送一条消息的应用程序，应完成下面3步：
 - (1) 创建一个Action类（控制器）控制与用户、模型和视图的交互。
 - (2) 创建一个服务器页面表示消息（视图）。
 - (3) 在struts.xml文件中建立Action类与视图的映射。

1. 创建Action动作类

- 动作是客户单击HTML页面中的超链接向Web服务器发送一个特定请求。动作类的execute()方法被执行并返回SUCCESS结果。Struts根据该结果返回一个视图页面（本例中是hellouser.jsp）。
- 下面是HelloUserAction类的定义。

1. 创建Action动作类

```
package com.action;

import com.opensymphony.xwork2.ActionSupport;

public class HelloUserAction extends
    ActionSupport {

    private String message;           // 动作属性
    // 省略getter方法和setter方法
    @Override
    public String execute() throws Exception {
        setMessage("Hello Struts User");
        return SUCCESS;
    }
}
```

2. 创建视图页面

- 程序12.2 index.jsp

- 该页面中使用了Struts的<s:url>标签。要使用Struts的标签，应该使用taglib指令导入标签库：

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

- 该指令指定了Struts 标签的prefix和uri属性值。Struts标签以前缀“s”开头，如<s:url>标签用来产生一个URL，它的action属性用来指定动作名，这里是hello。当用户点击该链接时将向容器发送hello.action请求动作。

2. 创建视图页面

- 创建下面的 JSP 页面 `hellouser.jsp` 来显示 `HelloUserAction` 动作类的 `message` 属性值，代码如下。
- 程序12.3 `hellouser.jsp`
- 页面中 `<s:property>` 标签显示 `HelloUserAction` 动作类的 `message` 属性值。通过在 `value` 属性中的 `message` 告诉 Struts 框架调用动作类的 `getMessage()`。

- index.jsp页面

```
<%@ page contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8"%>
```

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

```
<html><head><title>Basic Struts 2 Application -  
Welcome</title></head>
```

```
<body>
```

```
    <h3>Welcome To Struts 2!</h3>
```

```
    <p>
```

```
        <a href="<s:url action='hello'/">Hello User</a>
```

```
    </p>
```

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

- hellouser.jsp 页面

```
<%@ page contentType="text/html; charset=UTF-8"  
    pageEncoding="UTF-8"%>
```

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

```
<html><head>
```

```
<title>Hello User!</title></head>
```

```
<body>
```

```
    <h2><s:property value="message" /></h2>
```

```
</body>
```

```
</html>
```

3. 修改struts.xml配置文件

- 编辑struts.xml文件，在<package>元素中添加<action>定义，修改后的代码如下。

```
<action name="index">
    <result>/index.jsp</result>
</action>
<action name="hello"
    class="com.action.HelloUserAction "
    method="execute">
    <result name="success">/hellouser.jsp</result>
</action>
```

4. 程序的运行

- 访问index.jsp页面，当用户单击该页面中的“Hello User”链接。



[illegible]

6. 显示个性化信息

- 假设在index.jsp页面中通过表单提供用户名信息，在hellouser.jsp页面中显示用户名。
- 在index.jsp页面中添加下面的表单标签：

```
<s:form action="hello">
```

```
    <s:textfield name="userName" label="用户名" />
```

```
    <s:submit value="提交" />
```

```
</s:form>
```

6. 显示个性化信息

- HelloUserAction类必须定义一个名为userName的成员变量和一个名为setUserName()的public方法，Action对象会自动接收表单域的值。

```
private String userName;  
public String getUserName() {  
    return userName;  
}  
public void setUserName(String userName) {  
    this.userName = userName;  
}
```

6. 显示个性化信息

- 在execute()方法设置message属性值:

```
if (userName != null) {  
    setMessage(getMessage() + " " +  
    userName);  
}
```



12.1.5 Action动作类

- Struts 2应用程序可以完成的每一个操作都称之为一个动作。
 - 例如，点击一个超链接是一个动作，
 - 输入表单数据后点击提交按钮也是一个动作。
- 创建各种动作是Struts 2开发中最重要的任务。有些动作很简单，例如把控制权转交给一个JSP页面，而有些动作需要进行一些逻辑处理，这些逻辑需要写在动作类里。

12.1.5 Action动作类

- 处理这些动作使用动作类。动作类其实质就是Java类，它们可以有属性和方法，但必须遵守下面规则。
 - 每个属性都必须有一个getter方法和一个setter方法。动作属性的名字必须遵守与JavaBeans属性名同样的命名规则。动作的属性可以是任意类型，而不仅仅是String类型。

12.1.5 Action动作类

- 动作类必须有一个不带参数的构造方法。如果没有提供构造方法，Java编译器会自动提供一个默认构造方法。
- 每个动作类至少有一个方法供Struts 2在执行这个动作时调用。
- 一个动作类可以包含多个动作方法。在这种情况下，动作类可以为不同的动作提供不同的方法。例如，一个名为UserAction的动作类可以有login()和logout()方法，并让它们分别对应User_login和User_logout动作。

1. Action接口

- Struts 2定义了一个 `com.opensymphony.xwork2.Action` 接口，所有的动作类都可以实现该接口，该接口中定义了5个常量和一个 `execute()` 方法，如下所示：

1. Action接口

```
public interface Action {  
    public final static String SUCCESS = "success";  
    public final static String ERROR = "error";  
    public final static String INPUT = "input";  
    public final static String LOGIN = "login";  
    public final static String NONE = "none";  
    public String execute() throws Exception;  
}
```


2. ActionSupport类

- 编写动作类通常继承ActionSupport类，它是Action接口的实现类，该类还实现了Validateable接口、TextProvider等接口。
 - 设置校验错误的方法、返回校验错误的方法。
 - 设置和返回动作消息的方法。
 - 返回国际化信息的方法。

2. ActionSupport类

- ActionSupport类是Struts 2的默认动作处理类，即如果配置的Action没有指定class属性，系统自动使用ActionSupport类作为动作处理类。
- 在Struts 2中，动作类不一定必须实现Action接口，任何普通的Java对象（Plain Old Java Objects, POJO）只要定义execute()方法就可以作为动作类使用。

2. ActionSupport类

- 下面的MyAction类是最简单的动作类。

```
public class MyAction {  
    public String execute() throws Exception {  
        // 执行某些操作  
        return "success";  
    }  
}
```

- 注意，MyAction类没有实现任何接口和扩展任何类。动作执行将调用这里的execute()方法，该方法不带参数，返回一个String对象。

12.1.6 配置文件

- **struts.xml文件**主要用来建立动作Action类与视图的映射。
- 该文件是以<struts>为根元素的XML文件。
- 允许出现在<struts>和</struts>之间的直接子元素包括**package**、**constant**、**bean**和**include**，这些元素还可包含若干子元素。

12.1.6 配置文件

- struts.xml文件DTD的完整定义在struts2-core-2.3.8.jar文件中，文件名为struts-2.3.dtd。
- 下面对几个比较重要的元素进行讨论。

1. package元素

- <package>元素用来把动作组织成不同的包（package）。一个典型的struts.xml文件可以有一个或多个包。
- package元素的作用是对配置的信息进行逻辑分组。使用该元素可以将具有类似特征的action等配置信息定义为一个逻辑配置单元，这样可以避免重复定义。

1. package元素

- package元素的常用属性
 - **name**是指定该包的名称，其他包可使用此名称引用该包
 - **extends** 指定当前包继承哪一个已经定义的包
 - **namespace**为这个包指定一个URL映射地址
 - **abstract**指定当前包为抽象的，即该包中不能包含`action`的定义

1. package元素

- 在package中可以配置的信息包括
 - `action`
 - `result`
 - `interceptor`


```
<package name="example" namespace="/example"
extends="default">
  <action name="HelloWorld" class="example.HelloWorld">
    <result>/example/HelloWorld.jsp</result>
  </action>
  <action name="Login_*" method="{1}" class="example.Login">
    <result name="input">/example/Login.jsp</result>
    <result type="redirectAction">Menu</result>
  </action>
  <action name="*" class="example.ExampleSupport">
    <result>/example/{1}.jsp</result>
  </action>
</package>
```

2. action元素

- 用于定义一个动作。每个动作都必须有一个名字
- 属性：
 - name指定动作名称
 - class指定动作完整类名，缺省为 `ActionSupport` 类
 - method指定执行动作的方法名，缺省为 `execute()` 方法

2. action元素

- 如果动作有与之对应的动作类，则必须使用class属性指定动作类的完整名称。此外，还可以指定执行动作类的哪个方法。下面是一个例子。

```
<action name="Product_save"  
        class="com.action.Product"  
        method="save">
```

2. action元素

- 如果给出了class属性但没有给出method属性，动作方法的名字将默认为execute()。下面两个action元素的含义是等价的。

```
<action name="Emp_save"  
  class="com.action.EmployeeAction"  
  method="execute">
```

```
<action name="Emp_save"  
  class="com.action.EmployeeAction">
```

3. result元素

- 用来指定结果类型，即定义在动作完成后将控制权转到哪里。
- <result>元素对应动作方法的返回值。动作方法在不同的情况下可能会返回不同的值
- 一个<action>元素可能会有多个<result>元素，每个对应着动作方法的一种返回值。

3. result元素

- 比如，若某个方法有“success”和“input”两种返回值，就必须提供两个<result>元素。例如，下面的<action>元素包含两个<result>元素。

```
<action name="Product_save" class="com.action.Product"
  method="save">
  <result name="success" type="dispatcher">
    /jsp/Confirm.jsp
  </result>
  <result name="input" type="dispatcher">
    /jsp/Product.jsp
  </result>
</action>
```

3. result元素

- 如果省略<result>元素的name属性，其默认值是“success”，如果省略了type属性，默认结果类型是Dispatcher。下面两个<result>元素的含义是相同的。

```
<result name="success"  
    type="dispatcher">/jsp/Confirm.jsp</result>
```

```
<result>/jsp/Confirm.jsp</result>
```

4. global-results元素

- <package>元素可以包含一个<global-results>元素，其中包含一些通用的结果。如果某个动作在它的动作声明中不能找到一个匹配的结果，它将搜索<global-results>元素（如果有这个元素的话）。
- 下面是<global-results>元素的一个例子。

```
<global-results>
```

```
  <result name="error">
```

```
    /jsp/GenericErrorPage.jsp</result>
```

```
  <result name="login" type="redirect-action">
```

```
    login.jsp</result>
```

```
</global-results>
```


5. constant元素

- <constant>元素用来定义常量或覆盖 default.properties 文件里定义的常量。
- 如下所示的<constant>元素将把 struts.DevMode 项设置为 true。

```
<constant name="struts.DevMode" value="true">
```

6. include元素

- <include>元素的作用是包含其他的Struts 2配置文件。这样，通过<include>元素就可以轻松地把Struts 2的配置文件分解为多个文件。<include>元素是<struts>的直接子元素，下面是一个例子。

<struts>

<include file="module-1.xml" />

<include file="example.xml" />

</struts>

12.2 注册/登录系统

- 本节实现一个注册/登录系统。
- 按照MVC设计模式，可以将应用组件分成如下：
 - **模型层**包括存放用户信息的JavaBean类。
 - **持久层**包括数据库表、DAO类等。
 - **控制层**包括Action动作类。
 - **表示层**包括JSP页面。

12.2 注册/登录系统

- 12.2.1 定义持久化类
- 12.2.2 持久层实现
- 12.2.3 定义Action动作类
- 12.2.4 创建结果视图
- 12.2.5 修改struts.xml配置文件
- 12.2.6 运行应用程序

12.2.1 定义持久化类

- 为了封装表单数据，定义一个简单的User类，该类遵循JavaBeans规范，包含下面4个属性。

```
private String username;  
private String password;  
private int age;  
private String email;
```

- 程序12.4 User.java

12.2.2 持久层实现

- 用户数据存放在一个名为userinfo的数据表中，该表有username、password、age和email字段。

```
CREATE TABLE userinfo(  
    username varchar(20) PRIMARY KEY,  
    -- 用户名  
    password varchar(8) NOT NULL,  
    -- 口令  
    age int,  
    -- 年龄  
    email varchar(50) UNIQUE);  
-- Email地址
```

12.2.2 持久层实现

- 这里持久层实现使用Hibernate，需将库文件添加到WEB-INF/lib目录中，User类的映射文件User.hbm.xml如下。
- 程序12.5 User.hbm.xml

- 在配置文件hibernate.cfg.xml中增加下面一行：

```
<mapping  
    resource="com/model/User.hbm.xml"/  
>
```

12.2.3 定义Action动作类

- 下面的RegisterAction.java程序是一个动作类。在该类中声明了一个User类型的属性user，并为该属性定义了setter方法和getter方法。
- user对象与JSP页面表单域使用的user名匹配。
- [程序12.6 RegisterAction.java](#)

12.2.3 定义Action动作类

- 当表单提交时，Struts动作类首先使用User类的默认构造方法创建user属性对象，然后用表单域的值填充该user对象的每个属性，这个过程发生在execute()执行之前。
- 该类定义了register()和login()，分别用来处理注册和登录动作。在各自的方法体中创建一个Session对象，然后使用save(user())将user对象写入数据库，使用Query的list()检索数据库中是否存在user对象，从而实现注册和登录功能。

12.2.4 创建结果视图

- 为了将表单数据收集到User对象中，定义下面的页面register.jsp，其中包含一个表单用来接收用户输入数据。
- 程序12.7 register.jsp

12.2.4 创建结果视图

- 当用户点击“提交”按钮时系统执行Register动作，将表单数据提交给动作对象，因此需要在struts.xml文件中定义动作名称。
- 注意，4个输入域的name属性值对应于User类的4个属性，这里用对象名user来引用4个属性。当我们创建Action类处理该表单时，必须在Action类中指定该对象。

12.2.4 创建结果视图

- 页面中的<s:actionerror />和 <s:fielderror />标签用来显示动作错误和域校验的错误。该应用程序还包括登录页面login.jsp用来显示用户登录信息，代码如下。
- [程序12.8 login.jsp](#)
- success.jsp是注册成功显示的页面，代码如下。
- [程序12.9 success.jsp](#)

12.2.4 创建结果视图

- `success.jsp`是注册成功显示的页面，代码如下。
- 程序12.9 `success.jsp`
- 该页面通过<code>s:property</code>标签显示user对象的信息，它将调用User类的`toString()`输出结果。
- `welcome.jsp`页面用于显示登录成功欢迎消息，代码如下。
- 程序12.10 `welcome.jsp`

12.2.5 修改struts.xml配置文件

```
<action name="registerInput" >
```

```
    <result>/register.jsp</result>
```

```
</action>
```

```
<action name="loginInput" >
```

```
    <result>/login.jsp</result>
```

```
</action>
```

```
<action name="Register" class="com.action.RegisterAction"
```

```
    method="register">
```

```
    <result name="success">/welcome.jsp</result>
```

```
    <result name="error">/error.jsp</result>
```

```
</action>
```

```
<action name="Login" class="com.action.RegisterAction"
```

```
    method = "login">
```

```
    <result name="success">/success.jsp</result>
```

```
    <result name="error">/error.jsp</result>
```

12.2.6 运行应用程序

- 在index.jsp页面中添加下列代码定义两个动作

```
<p><a href="<s:url action='registerInput' />">用户注册</a></p>
```

```
<p><a href="<s:url action='loginInput' />">用户登录</a></p>
```

- 在index.jsp页面中单击“用户注册”链接，打开register.jsp页面，如图12-4所示。在该页面中输入用户信息，单击“注册”按钮，则显示如图12-5所示页面。

12.2.6 运行应用程序

用户注册

http://localhost:8080/basicstruts2/registerInput.action

注册一个新用户

用户名: 王小明

口令: ●●●●●●●●

年龄: 20

Email地址: wangxiao@163.com

注册

注册成功页面

http://localhost:8080/basicstruts2/Register.action

注册成功!

用户名: 王小明 口令: wangxiao 年龄: 20 Email: wangxiao@163.com

[返回](#) [首页](#)

12.3 OGNL

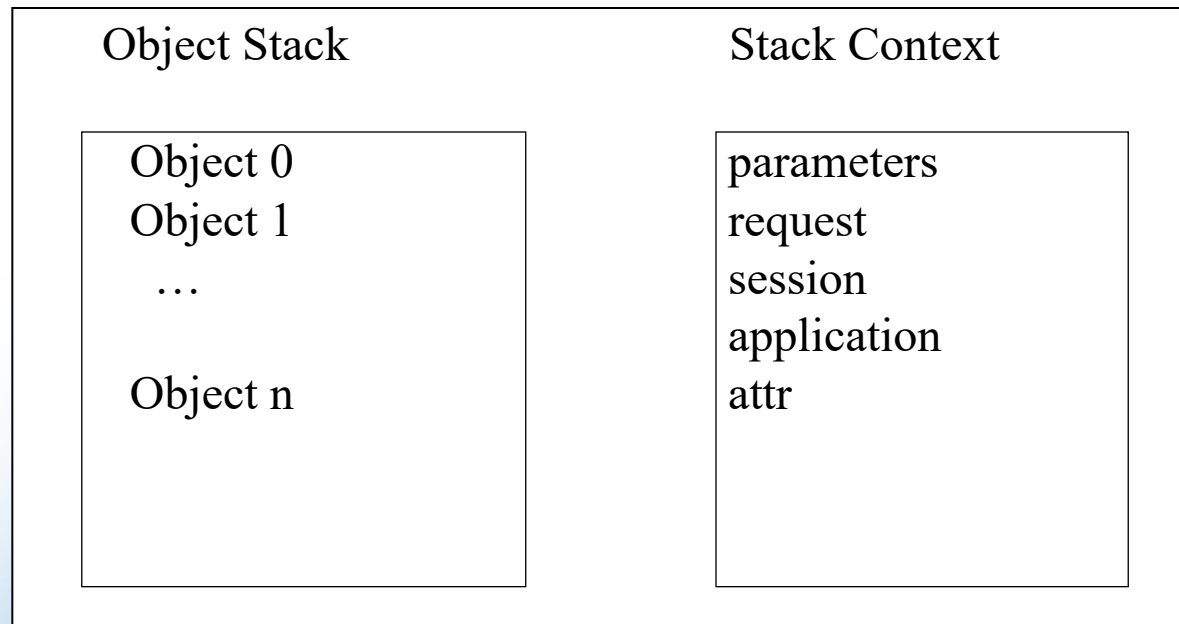
- OGNL (Object-Graph Navigation Language) 称为对象-图导航语言，它是一种简单的、功能强大的表达式语言。
- 使用OGNL表达式语言可以访问存储在ValueStack和ActionContext中的数据。

12.3.1 ValueStack栈

- 对应用程序的每一个动作，Struts在执行相应的动作方法前会先创建一个ValueStack对象，称为**值栈**。ValueStack用来保存该动作对象及其属性。
- JSP页面能够访问ValueStack
-

12.3.1 ValueStack栈

- 在ValueStack栈的内部有两个逻辑组成部分，分别是Object Stack和Stack Context，如图12.6所示。



12.3.2 读取Object Stack中对象属性

- 要读取栈顶对象并检索对象的属性，则标签必须用类似下面的形式message可直接写成message的形式

```
<s:property value="#{message}" />
```

- 还可以使用下面的语法访问动作类的getMessage()方法：

```
<s:property value="#{getMessage}" />
```

示例

- 定义SampleAction动作类

```
public class SampleAction {  
    private String message;  
    private UserBean user =  
        new UserBean();  
  
    {  
        user.setUsername("王小明");  
    }  
  
    public String execute() {  
        setMessage("世界，你好！");  
        return "success";  
    }  
}
```

示例

- 在struts.xml文件中使用下面<action>元素定义动作。

```
<action name="sample"  
    class="com.action.SampleAction"  
        method="execute">  
    <result  
        name="success">/sample.jsp</result>  
</action>
```

示例

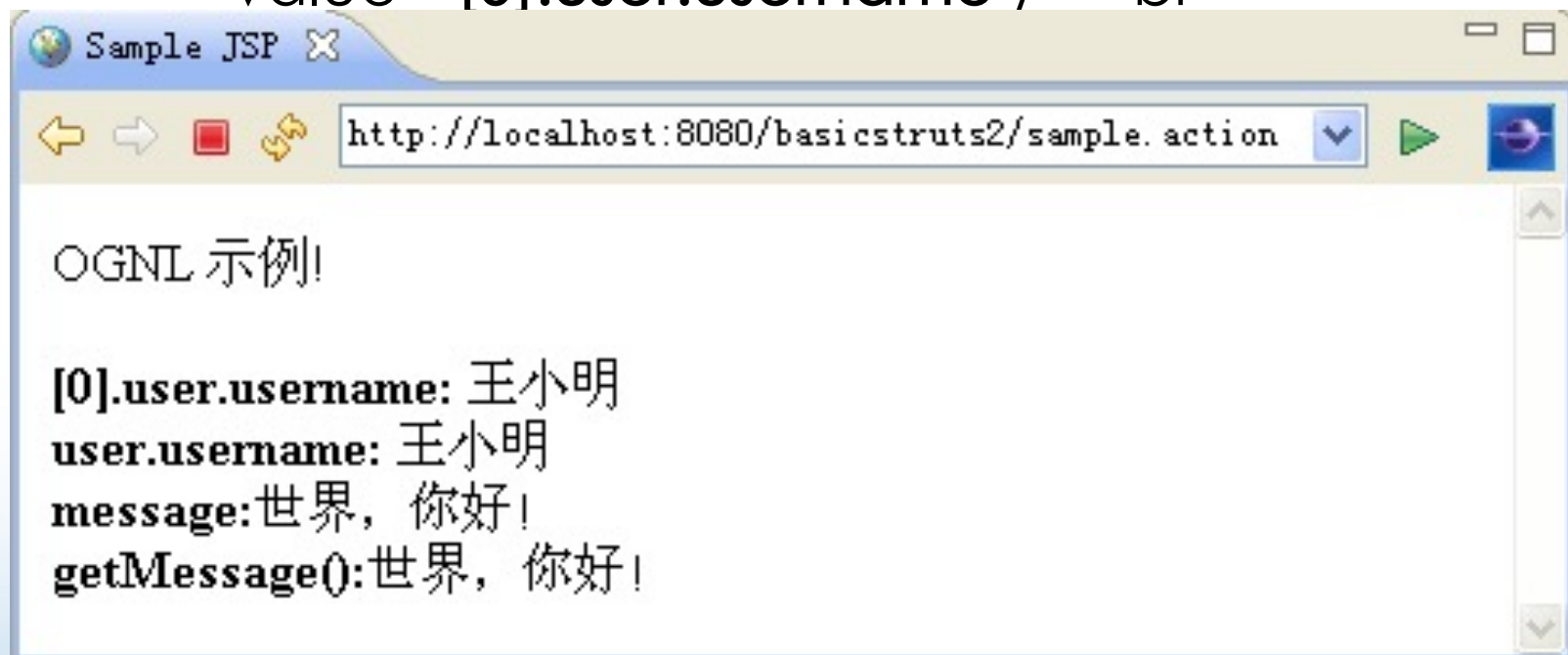
- 在index.jsp页面中添加下面代码定义一个超链接引发sample动作。

```
<a href="<s:url action='sample'/'>">  
    Sample JSP</a>
```

示例

sample.jsp

```
<b>[0].user.username:</b> <s:property  
    value="[0].user.username"/> <br>
```



12.3.3 读取Stack Context对象的属性

- 在Stack Context中包含下列对象：
application、session、request、
parameters、attr。
- 这些对象都是Map类型的对象，可在其中
存储“键/值”对数据。

12.3.3 读取Stack Context对象的属性

- **application**中包含当前应用的Servlet上下文属性
- **session**中包含当前会话级属性
- **request**中包含当前请求级属性
- **parameters**中包含当前请求的请求参数
- **attr**用于在request、session和application作用域中查找指定的属性。

12.3.3 读取Stack Context对象的属性

- 要访问Stack Context中的对象需要给OGNL表达式加上一个前缀字符“#”，可以使用以下几种形式之一：

`#object.propertyName`

`#object['propertyName']`

`#object["propertyName"]`

这里object为上述5个对象之一，propertyName为对象中的属性名。例如：

`<s:property value="#application.userName" />`

12.3.5 使用OGNL访问数组元素

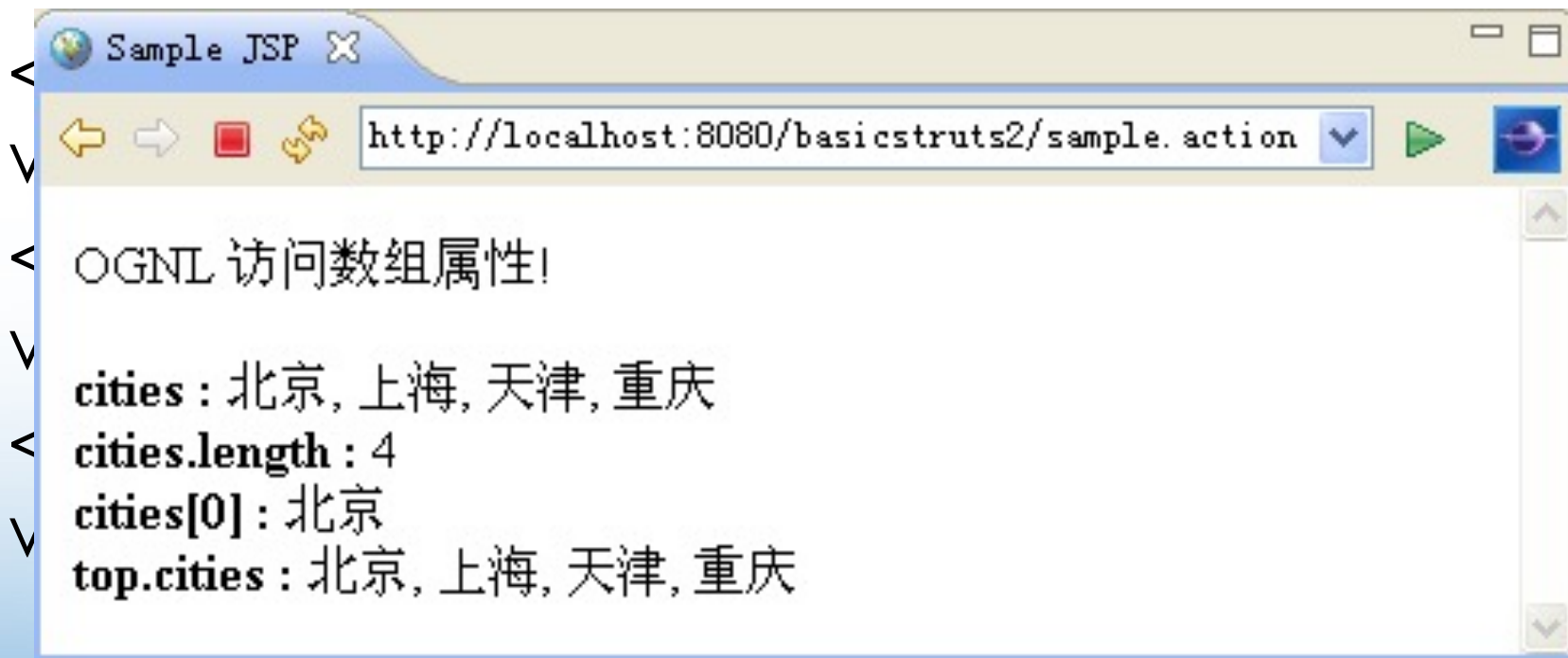
- 若动作类SampleAction中声明一个String数组属性，在JSP页面中可以使用<s:property>标签访问。

```
private String[] cities;  
public String execute() {  
    cities = new String[]{"北京","上海","天津","重庆"  
};  
    return "success";  
}
```

12.3.5 使用OGNL访问数组元素

- 在JSP页面中可以使用OGNL按如下方式访问数组元素。

```
<b>cities :</b> <s:property value="cities"/>  
<br>
```



12.3.6 使用OGNL访问List类型的属性

```
private List<String> fruitList = new  
ArrayList<String>();  
{  
    fruitList.add("苹果");  
    fruitList.add("橘子");  
    fruitList.add("香蕉");  
}  
public String execute(){  
    return "success";  
}
```

12.3.6 使用OGNL访问List类型的属性

- 在JSP页面中可以使用OGNL按如下方式访问ArrayList的元素。

```
<b>fruitList:</b><s:property value="fruitList"/>  
<br>
```

```
<b>fruitList.size:</b><s:property  
    value="fruitList.size"/> <br>
```

```
<b>fruitList[0]:</b><s:property  
    value="fruitList[0]"/> <br>
```

12.3.7 使用OGNL访问Map类型的属性

```
private Map<String,String> countryMap =  
    new HashMap<String,String>();  
{  
    countryMap.put("China", "北京");  
    countryMap.put("American", "纽约");  
    countryMap.put("Australia", "堪培拉");  
}  
public String execute(){  
    return "success";  
}
```


12.3.7 使用OGNL访问Map类型的属性

- 在JSP页面中可以使用OGNL按如下方式访问Map的元素。

```
<b>countryMap:</b><s:property  
    value="countryMap"/><br>
```

```
<b>countryMap.size:</b><s:property  
    value="countryMap.size"/><br>
```

```
<b>countryMap[1]:</b><s:property  
    value="countryMap['China']"/> <br>
```

12.4 Struts 2常用标签

- Struts 2框架提供了一个标签库使得Web应用程序可以很容易地在页面中引用动态数据，创建动态响应。
- 有些标签模仿标准的HTML标签，还有些标签用于创建非标准的控件。
- Struts 2的标签可以分为两大类：
 - **通用标签**
 - **用户界面（UI）标签**

12.4 Struts 2常用标签

标签分类		标 签
通用标签	数据标签	a、action、bean、date、debug、i18n、include、param、push、set、text、url、property
	控制标签	if、elseif、else、append、generator、iterator、merge、sort、subset
UI标签	表单标签	checkbox、checkboxlist、combobox、doubleselect、file、hidden、label、optiontransfersselect、optgroup、password、radio、reset、select、submit、textarea、textfield、token、updownselect
	非表单标签	actionerror、actionmessage、component、fielderror、table、
	Ajax标签	a、autocompleter、datetimepicker、div、head、submit、tabbedPanel、tree、treenode

12.4.1 通用标签

- 1. `<s:property>` 标签
- 2. `<s:param>` 标签
- 3. `<s:bean>` 标签
- 4. `<s:set>` 标签
- 5. `<s:push>` 标签
- 6. `<s:url>` 标签
- 7. `<s:action>` 标签
- 8. `<s:date>` 标签
- 9. `<s:include>` 标签
- 10. `<s:debug>` 标签

1. <s:property> 标签

- 用于在页面中输出一个动作属性值。
- 例如，下面标签将输出customerId动作属性的值。

```
<s:property value="customerId" />
```

- 下面这个<s:property>标签将输出会话作用域中名为userName的属性值。

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

1. `<s:property>` 标签

- 如果没有给出value属性，将输出ValueStack栈顶对象的值。
- 通常，EL语言可以提供更简洁的语法。
- 例如，下面的EL表达式同样可以输出customerId动作属性的值。

`${customerId}`

2. <s:param> 标签

- <s:param> 标签用于把一个参数传递给包含它的标签（如<s:bean>、<s:url>等）。
- 它有两个属性：
 - name, 值为参数名
 - value, 值为参数值。

```
<s:param name="userName"  
value="userName" />
```

```
<s:param name="userName"  
value="%{userName}" />
```

2. <s:param> 标签

- 如果要传递一个String类型的字符串作为参数值，必须把它用单引号括起来。例如：

```
<s:param name="empName" value="John  
Smith"></s:param>
```

- 也可以将value属性值写在<s:param>标签的开始标签和结束标签之间，如下所示：

```
<s:param name="empName">John  
Smith</s:param>
```


3. `<s:bean>` 标签

- `<s:bean>` 标签用于创建JavaBean实例，并把它压入Value Stack栈的Stack Context。这个标签的功能与JSP的`<jsp:useBean>`动作很相似。
- 下面定义一个ConverterBean用来实现摄氏温度和华氏温度的相互转换。

3. <s:bean> 标签

```
public class ConverterBean{  
    private double celcius;  
    private double fahrenheit;  
  
    public double getCelcius(){  
        return (fahrenheit - 32) * 5 / 9;}  
    public void setCelcius(double celcius){  
        this.celcius = celcius;}  
    public double getFahrenheit(){  
        return celcius * 9 / 5 + 32;}  
    public void setFahrenheit(double fahrenheit){  
        this.fahrenheit = fahrenheit;}  
}
```

3. <s:bean> 标签

下面的tagDemo.jsp页面中使用了<s:bean>和<s:param>标签。

```
<body>
```

```
<p>Bean Tag Example</p>
```

```
<s:bean name="com.model.ConverterBean" id="converter">
```

```
<s:param name="celcius">37</s:param>
```

```
</s:
```

```
378 Bean Tag Example
```

```
>&lt; 37° C=98.6° F
```

```
</b
```



heit"/

4. <s:set> 标签

- <s:set>标签用来在指定作用域中定义一个属性并为其赋值，然后将其存储到Stack Context中。
- 使用<s:set>标签定义popLanguage变量并赋值，然后访问该变量。name属性指定变量名，value属性指定变量值。

```
<s:set name="popLanguage" value="%{'Java'}"/>
```

```
Popular Language is: <s:property  
value="#popLanguage"/>
```

5. `<s:push>` 标签

- `<s:push>` 标签与 `<s:set>` 标签类似，区别是 `<s:push>` 标签把一个对象压入 `ValueStack` 而不是 `Context Map`。
- `<s:push>` 标签的另一个特殊的地方是，它的起始标签把一个对象压入栈，结束标签将弹出该对象。
- `<s:push>` 标签只有一个 `value` 属性，它指定将被压入 `ValueStack` 栈中的值。

5. <s:push> 标签

- 假设有一名为Employee的JavaBean类，该类有name和age两个属性。在JSP中可使用下列代码创建一个bean实例并将其压入ValueStack。

```
<s:bean name="com.model.Employee" var="empBean">
```

```
姓名: <s:property value="#empBean.name"/><br/>
```

```
年龄: <s:property value="#empBean.age"/>
```

```
</s:bean>
```

```
<hr/>
```

```
<s:push value="#empBean">
```

```
姓名: <s:property value="name"/><br/>
```

```
年龄: <s:property value="age"/>
```

6. <s:url> 标签

- <s:url>标签用来创建一个超链接，指向其他Web资源，尤其是本应用程序的资源。例如：

```
<p><a href="<s:url action='hello'/'>">Hello World</a></p>
```

- 该标签通过action属性指定引用的资源。当程序运行时，将鼠标指向链接，可以看到链接的目标是hello.action，它相对于Web应用程序的根目录。

6. <s:url> 标签

- 在<s:url>标签内可以使用<s:param>标签为URL提供查询串。例如：

```
<s:url action="hello" var="helloLink">
```

```
  <s:param name="userName">Bruce Phillips</s:param>
```

```
</s:url>
```

```
<p><a href="{helloLink}">Hello Bruce Phillips</a></p>
```

- 这里，使用<s:param>为请求提供一个查询参数，userName为参数名，标签内的值为参数值。

7. `<s:action>` 标签

- `<s:action>` 标签用于在 JSP 页面中直接调用一个 Action。

8 <s:date> 标签

- <s:date>标签用来对Java语言的一个Date对象进行格式化。用户可以指定一种输出格式（如，“dd/MM/yyyy hh:mm”），还可以产生易读的格式（如，“2小时，14分钟”）。

```
private Date currentDate;  
public String execute() throws Exception{  
    setCurrentDate(new Date());  
    return SUCCESS;  
}
```

9 <s:include> 标签

- `<s:include>` 标签用于将一个JSP页面，或者一个Servlet的输出包含到本页面中。该标签只有一个必须的value属性，用于指定需要包含的JSP页面或Servlet。
- 在 `<s:include>` 标签体中还可以使用 `<s:param>` 子标签为被包含的JSP页面或Servlet传递参数。

9 <s:include> 标签

<p>Include Tag (Data Tags) Example!</p>

<s:include value="included_file.jsp" />

<s:include value="included_file.jsp">

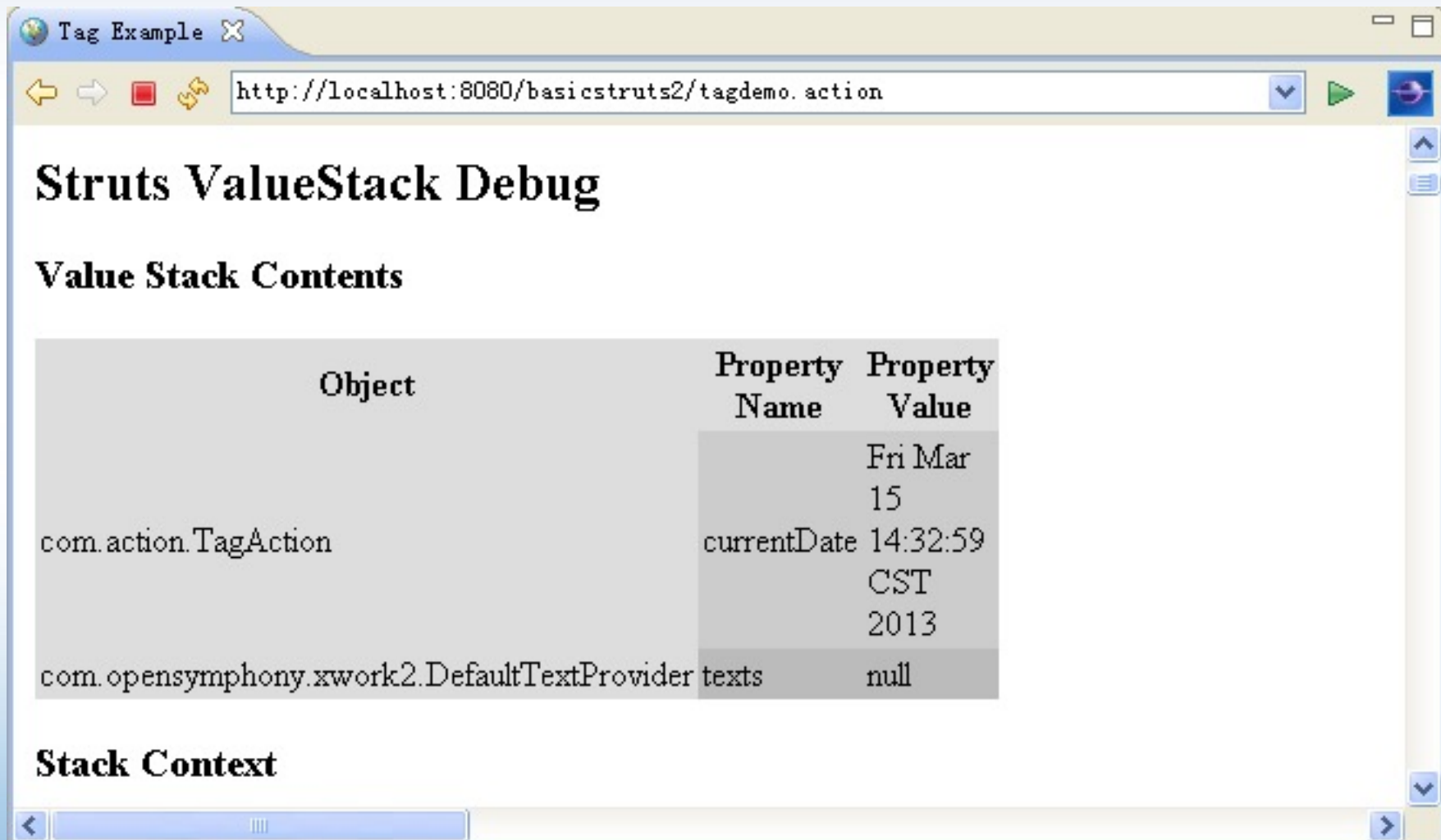
<s:param

name="title">Hello,World!</s:param>

</s:include>

10. <s:debug> 标签

- 在JSP页面中可以使用<s:debug />标签，它



The screenshot shows a web browser window titled "Tag Example" with the address bar displaying "http://localhost:8080/basicstruts2/tagdemo.action". The page content is titled "Struts ValueStack Debug" and includes a section "Value Stack Contents" which displays a table of the current value stack. The table has three columns: "Object", "Property Name", and "Property Value". The first object is "com.action.TagAction" with properties "currentDate" (value: "Fri Mar 15 14:32:59 CST 2013") and "texts" (value: "null"). The second object is "com.opensymphony.xwork2.DefaultTextProvider" with property "texts" (value: "null"). Below the table is a section titled "Stack Context" which is currently empty.

Object	Property Name	Property Value
com.action.TagAction	currentDate	Fri Mar 15 14:32:59 CST 2013
	texts	null
com.opensymphony.xwork2.DefaultTextProvider	texts	null

12.4.2 控制标签

- 1.<s:if>, <s:else>和<s:elseif> 标签
- 2.<s:iterator> 标签
- 3.<s:append> 标签
- 4.<s:merge> 标签
- 5.<s:generator> 标签
- 6.<s:sort> 标签

1. <s:if>, <s:else>和<s:elseif> 标签

- 这3个标签用来进行条件测试，它们的用途与Java语言中的if，else和else if结构类似。<s:if>和<s:elseif>标签必须带一个test属性，用来设置测试条件。
- 例如，下面这个<s:if>标签用来测试name请求参数是否为空值null。

```
<s:if test="#parameters.name ==null">
```

1. <s:if>, <s:else>和<s:elseif> 标签

- 下面例子使用<s:if>标签测试会话属性 `loggedIn` 是否存在。若不存在显示一个登录表单，若存在显示欢迎信息。

```
public String execute(){
    if(username !=null && username.length() > 0
        && password != null && password.length()> 0){
        ServletActionContext.getContext().
            getSession().put("loggedIn", true);
    }
    return SUCCESS;
}
```


1. <s:if>, <s:else>和<s:elseif> 标签

```
<s:if test="#session.loggedIn ==null">
```

```
  <h3>请输入用户名和口令</h3>
```

```
  <s:form>
```

```
    <s:textfield name="username" label="用户名" />
```

```
    <s:password name="password" label="口令" />
```

```
    <s:submit value="登录" />
```

```
  </s:form>
```

```
</s:if>
```

```
<s:else>
```

```
  Welcome <s:property value="username" />
```

```
</s:else>
```

2. <s:iterator> 标签

- <s:iterator>标签可以遍历一个数组、一个Collection或一个Map对象并把其中的每一个元素压入和弹出ValueStack栈。

表12.10 <s:iterator>标签的属性

属性名	类型	说 明
var	String	指定一个变量存放这个可遍历对象当前元素
value	String	将被遍历的可遍历对象
status	IteratorStatus	状态对象
id	Strng	功能同var属性

2. `<s:iterator>` 标签

- `<s:iterator>` 标签在开始执行时，会先把 `org.apache.struts2.views.jsp.IteratorStatus` 类的一个实例压入 `Context Map` 并在每次遍历时更新它。
- 可以将一个指向这个 `IteratorStatus` 对象的变量赋给 `status` 属性。

2. <s:iterator> 标签

表12.11 IteratorStatus对象的属性

属性名	类型	说 明
index	int	每次遍历的下标值，从0开始
count	int	当前遍历的下标值
first	boolean	当前遍历的是否是第一个元素
last	boolean	当前遍历的是否是最后一个元素
even	boolean	如果count属性值是偶数，返回true
odd	boolean	如果count属性值是奇数，返回true
modulus	int	这个属性需要一个参数，它的返回值是count属性值除以输入参数的余数

2. `<s:iterator>` 标签

- 下面的例子在 `IteratorAction` 动作类中定义了一个 `List` 属性和一个 `Map` 属性，并向其中添加了一些元素。 `iteratorTag.jsp` 页面演示了如何使用 `<s:iterator>` 标签访问这些集合对象的元素。
- 程序12.15 `IteratorAction.java`
- 下面是 `iteratorTag.jsp` 页面代码。
- 程序12.16 `iteratorTag.jsp`

2. <s:iterator> 标签

```
<s:iterator value="fruit" status="status">
  <s:property />
  <s:if test="!#status.last">, </s:if>
  <s:else><br></s:else>
</s:iterator>
```



2. <s:iterator> 标签

- <s:iterator>标签的value属性值也可以通过常量或使用<s:set>标签指定值，例如：

```
<s:iterator  
  value="{ 'one','two','three','four' }">
```

```
<s:property />
```

```
</s:iterator>
```

2. <s:iterator> 标签

```
<s:set name="os"  
value="{ 'Windows','Linux','Solaris' }" />
```

```
<s:iterator value="#os" status="status">  
  <s:property /><s:if test="!#status.last"> ,</s:if>  
</s:iterator>
```


3. `<s:append>` 标签

- `<s:append>` 标签用于将多个集合对象拼接起来，形成一个新的集合。这样就可以通过一个 `<s:iterator>` 标签实现对多个集合的迭代。
- 使用 `<s:append>` 标签需要指定一个 `var` 属性，该属性值用来存放拼接后生成的集合对象，新集合被放入 `Stack Context` 中。

3. `<s:append>` 标签

- `<s:append>` 标签可以带多个 `<s:param>` 标签，每个 `<s:param>` 标签用来指定一个需要拼接的集合。
- 子集合中的元素是以追加的方式拼接，即后面集合的元素追加到前面集合元素的后面。

3. <s:append> 标签

```
<s:set var="myList" value="{ 'one','two','three' }" />
```

```
<!-- 拼接3个集合 -->
```

```
<s:append var="newList">
```

```
  <s:param value="{ 'Operating System','Data Structure',  
'Java Programming' }" />
```

```
  <s:param value="#myList" />
```

```
  <s:param value="fruit" /> <!-- 动作类的属性 -->
```

```
</s:append>
```

3. <s:append> 标签

```
<table border="1" width="260">  
  <s:iterator value="#newList" status="status" id="elem">  
    <tr>  
      <td><s:property value="#status.count" /></td>  
      <td><s:property value="elem" /></td>  
    </tr>  
  </s:iterator>  
</table>
```

4. `<s:merge>` 标签

- `<s:merge>` 标签是将多个集合的元素合并，该标签与 `<s:append>` 标签类似。
- 新集合的元素完全相同，但不同的是，`<s:merge>` 标签是以交叉的方式合并集合元素。

5. `<s:generator>` 标签

- `<s:generator>` 标签可以将指定字符串按指定分隔符分割成多个子串，临时生成的子串可以使用 `<s:iterator>` 标签迭代输出。
- 在该标签体内，生成的集合位于 `ValueStack` 的顶端，一旦该标签结束，该集合将被移出 `ValueStack`。
- `<s:generator>` 标签的作用有点类似于 `String` 类的 `split()` 方法，但它比 `split()` 方法的功能更强大。

5. <s:generator> 标签

表12.12 <s:generator>标签的属性

属性名	类型	说 明
val	String	指定被解析的字符串
seperator	String	指定各元素之间的分隔符
count	Integer	可遍历对象最多能够容纳的元素个数
var	String	用来引用新生成的可遍历对象的变量
converter	Converter	指定一个转换器

5. <s:generator> 标签

```
<s:generator val="%{'三星,诺基亚,摩托罗拉,小米'}"  
separator="," >
```

```
<ul>
```

```
<s:iterator><li><s:property/></li></s:iterator>
```

```
</ul>
```

```
</s:generator>
```

```
<s:generator val="%{'奥迪,丰田,宝马,比亚迪'}"  
separator="," id="cars" count="3" >
```

```
</s:generator>
```

```
<s:iterator value="#attr.cars">
```

```
<s:property />
```

```
</s:iterator>
```


6. <s:sort> 标签

- <s:sort>用来对一个可遍历对象里的元素进行排序。表12.13列出了它的属性。

属性名	类型	默认值	说 明
comparator	java.util.Comparator		指定在排序过程中使用的比较器
source	String		将对之进行排序的可遍历对象
var	String		用来引用因排序而新生成的可遍历对象的变量

6. `<s:sort>` 标签

- 下面的例子在SortTagAction类中定义了一个ArrayList对象存放Student对象，一个myComparator的比较器对象，使用该对象对学生集合进行排序。
- [程序12.17 SortTagAction.java](#)
- 在JSP页面中使用`<s:sort>`标签对学生对象使用指定的比较器进行排序，结果可存入一个变量中，然后使用`<s:iterator>`标签迭代输出。
- [程序12.18 sortDemo.jsp](#)

12.4.3 表单UI标签

- 表单UI标签主要用来在HTML页面中显示数据。UI标签的使用非常简单。UI标签可以根据选定的主题自动生成HTML代码。默认情况下，使用XHTML主题，该主题使用表格定位表单元素。

1. 表单标签的公共属性

- 在HTML语言中，表单中的元素拥有一些通用的属性，如 `id` 属性、`name` 属性以及JavaScript中的事件等。
- 与HTML中相同，Struts 2提供的表单标签也存在通用的属性，而且这些属性比较多。

1. 表单标签的公共属性

属性名	数据类型	说 明
name	String	指定HTML的name属性
value	String	指定表单元素的值
cssClass	String	用来呈现这个元素的CSS类
cssStyle	String	用来呈现这个元素的CSS样式
title	String	指定HTML的title属性
disabled	String	指定HTML的disabled属性
tabIndex	String	指定HTML的tabindex属性

1. 表单标签的公共属性

label	String	指定一个表单元素在xhtml和ajax主题里的行标
labelPosition	String	指定一个表单元素在xhtml和ajax主题里的行标位置。可能取值为top和left（默认值）
required	boolean	在xhtml主题里，这个属性表明是否要给当前行标加上一个星号*
theme	String	指定主题的名字
template	String	指定模板的名字
onclick	String	指定JavaScript的onclick属性
onmouseover	String	指定JavaScript的onmouseover属性
onchange	String	指定JavaScript的onchange属性
tooltip	String	指定浮动提示框的文本

2. `<s:form>` 标签

- `<s:form>` 标签用来创建表单，它使得创建输入表单更容易。Struts 2 表单标签模拟普通的表单标签，每个 `<s:form>` 标签带有多个属性，`action` 属性用来指定动作。

3. <s:textfield>和<s:password>标签

- <s:textfield>标签用来生成HTML的单行输入框，
- <s:password>标签用来生成HTML的口令输入框，这两个标签的公共属性如表12.15所示。

属性名	类型	说 明
maxlength	int	指定文本框能容纳的最大字符数
readonly	boolean	指定文本框的内容是否只读，默认为false
size	int	指定文本框的大小

3. <s:textfield>和<s:password>标签

- 下面是一段简单的表单代码。

```
<s:form action="login">  
    <s:textfield name="userName"  
label="用户名" />  
    <s:submit value="提交" />  
</s:form>
```

3. `<s:textfield>`和`<s:password>`标签

- Struts 2 表单标签最终都转换成HTML标准标签。查看页面的源文件，`<s:form>`标签转换后的代码

4. <s:textarea>标签

- <s:textarea>标签用来生成HTML的文本区，该标签的常用属性如表12.16所示。

属性名	类型	默认值	说 明
rows	integer		指定文本区的行数
cols	integer		指定文本区的列数
readonly	boolean	false	指定文本区内容是否只读
wrap	boolean		指定文本区内容是否回绕

4. <s:textarea>标签

- 例如，下面代码生成一个8行35列的文本区。

```
<s:textarea name="description"  
    label="简历："   
    rows="8" cols="35" />
```

5. <s:submit>和<s:reset>标签

- <s:submit>标签用来生成HTML的提交按钮。根据其type属性的值，这个标签可以有3种显示效果。下面是type属性的合法取值。
 - input：把标签呈现为<input type="submit" ... />
 - button：把标签呈现为<button type="submit" ... />
 - image：把标签呈现为<input type="image" ... />

5. <s:submit>和<s:reset>标签

- <s:reset>标签用来生成HTML的重置按钮。根据其type属性的值，这个标签可以有两种显示效果。下面是type属性的合法取值。
 - input：把标签呈现为 - button：把标签呈现为<button type="reset" ... />

5. <s:submit>和<s:reset>标签

表12.17 <s:submit>标签和<s:reset>标签的属性

属性名	类型	说 明
value	String	指定提交或重置按钮上显示的文字
action	String	指定HTML的action属性
method	String	指定HTML的align属性
type	String	指定按钮的屏幕显示类型，默认值为input

6. <s:checkbox>标签

- <s:checkbox>标签用来生成HTML的复选框元素。该标签返回一个布尔值，若被选中返回“true”，否则返回“false”。
- 例如：

```
<s:checkbox name="mailingList" label="是否加入邮件列表?" />
```
- <s:checkbox>标签还有一个非常有用的属性 `fieldValue`，它指定的值将在用户提交表单时作为被选中的实际值发送到服务器。`fieldValue`属性可以用来发送一组复选框的被选中值。

7. <s:radio>标签

- <s:radio>标签用来生成HTML的**单选按钮组**，单选按钮的个数与程序员通过该标签的list属性提供的选项个数相同。通常使用<s:radio>标签实现“多选一”的应用。
- 除了具有表单标签共同的属性外，<s:radio>标签还提供了如表12.18所示的常用属性。

7. <s:radio>标签

表12.18 <s:radio>标签的属性

属性名	类型	说 明
list	String	指定选项来源的可遍历对象
listKey	String	指定选项值的对象属性
listValue	String	指定选项行标的对象属性

7. <s:radio>标签

- 例如:

```
<s:radio name="gender" label="性别"  
list="{ '男', '女' }" />
```

- list、listKey和listValue属性对<s:radio>标签、<s:combobox>标签、<s:select>标签、<s:checkboxlist>标签和<doubleselect>标签来说非常重要，因为它们可以帮助程序员更有效率地管理和获取这些标签的选项。

8. <s:checkboxlist>标签

- <s:checkboxlist>标签将呈现为一组复选框，它的属性如表12.20所示。

属性名	类型	默认值	说 明
list	String		指定选项来源的可遍历对象
listKey	String		指定选项值的对象属性
listValue	String		指定选项行标的对象属性

8. <s:checkboxlist>标签

- <s:checkboxlist>标签将被映射到一个字符串数组或一个基本类型的数组。如果它提供的复选框一个也没被选中，相应的属性将被赋值为一个空数组而不是空值。
- 下面的代码演示了<s:checkboxlist>标签的用法。

```
<s:checkboxlist name="language"  
    list="langList" label="精通语言" />
```

9. <s:select>标签

- <s:select>标签用来生成HTML的下拉列表框元素，它的属性如表12.19所示。

9. <s:select>标签

属性名	类型	默认值	说 明
list	String		指定选项来源的可遍历对象
listKey	String		指定选项值的对象属性
listValue	String		指定选项行标的对象属性
headerKey	String		指定选项列表中第一个选项的键
headerValue	String		指定选项列表中第一个选项的值
emptyOption	boolean	false	指定是否在标题下面插入一个空白选项
multiple	boolean	false	指定是否允许多重选择
size	integer		指定同时显示在页面里的选项个数

9. <s:select>标签

- 下面是<s:select>标签一个例子:

```
<s:select name="city" list="cityList"  
listKey="cityId" listValue="cityName"  
headerKey="0" headerValue="城市"  
label="请选择城市" />
```


12.4.4 实例：UI标签使用

- 下面实例演示了几个UI标签的使用，动作类RegisterAction的代码如下。

12.4.4 实例：UI标签使用

注册页面

http://localhost:8080/basicstruts2/populateRegister.action

用户名: 张大海

口令: ●●●●●●

性别: ☐ 男 ☒ 女

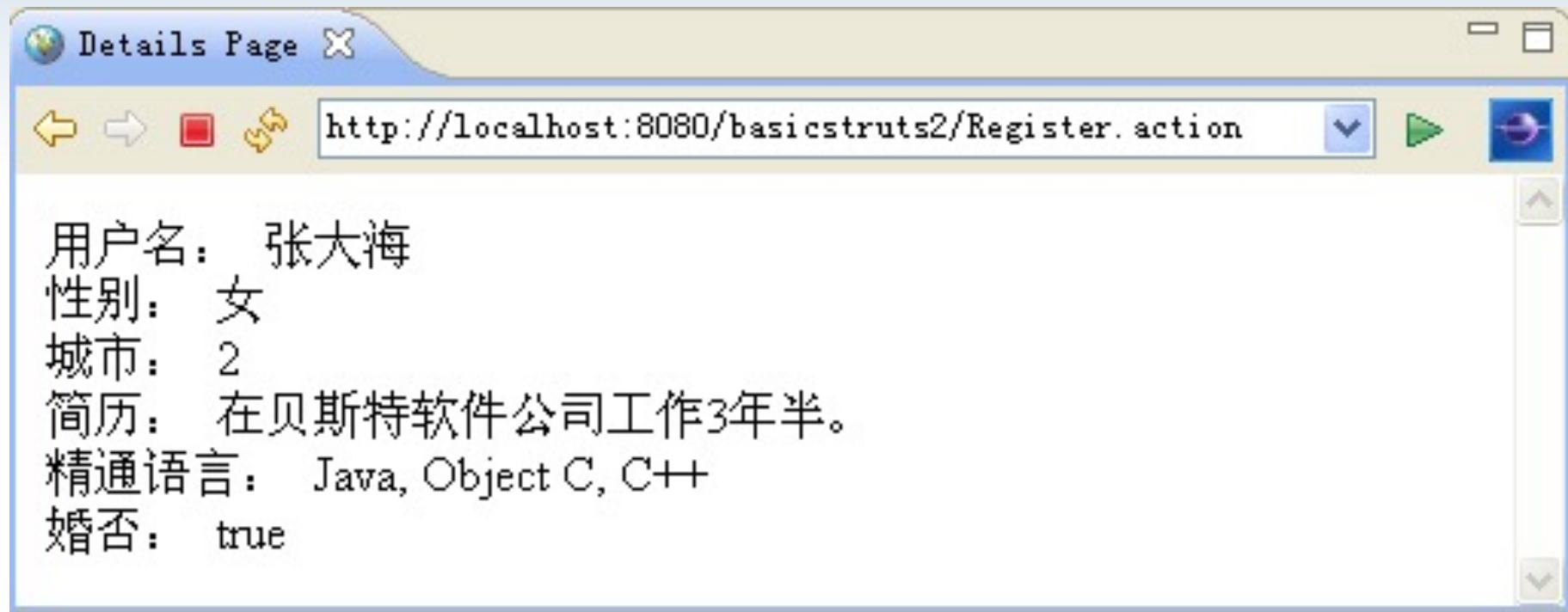
请选择城市: 上海

简历: 在贝斯特软件公司工作3年半。

精通语言: ☒ Java ☐ .Net ☒ Object C ☒ C++
☒ 婚否?

提交

12.4.4 实例：UI标签使用



12.4.4 实例：UI标签使用

- 下面实例演示了几个UI标签的使用，动作类RegisterAction的代码如下。

RegisterAction

- 下面是register.jsp页面代码。

register.jsp

- success.jsp页面的代码如下。

success.jsp

12.4.4 实例：UI标签使用

- 在struts.xml文件中添加下面的action定义：

```
<action name="*Register" method="{1}"
        class="com.action.RegisterAction">
    <result
name="populate">/register.jsp</result>

    <result
name="input">/register.jsp</result>
    <result
name="success">/success.jsp</result>
</action>
```

12.4.5 模板与主题

- Struts 2 标签库的每一个标签都将呈现为一个或多个HTML元素。Struts 2 允许我们选择这些元素以何种方式呈现。
- 例如，在默认情况下，`<s:form>` 标签将呈现为HTML的一个form元素和一个table元素。每一种输入标签（如 textfield、checkbox和submit）都将呈现为一个带标号的输入元素，这个输入元素将被包含在一个tr元素和一个td元素内。

12.4.5 模板与主题

- 默认情况下表单<code>s:form>标签被排版成表格的形式，但在某些场合，我们可能希望按照自己的想法来进行排版。例如，如果希望<code>s:textfield>元素呈现为一个单独的<code>input>标签，而不是一个包含在<code>tr>和<code>td>标签中的输入元素。

12.4.5 模板与主题

```
<s:form>
```

```
  <s:textfield label="用户名" />
```

```
  <s:submit />
```

```
</s:form>
```


12.4.5 模板与主题

```
<form id="test" name="test"
action="/chap12/test.jsp" method="post">
<table class="wwFormTable">
  <tr><td class="tdLabel"><label for="test_"
class="label">用户名:</label></td>
    <td><input type="text" name=""
id="test_" /></td>
  </tr>
  <tr><td colspan="2"><div align="right">
    <input type="submit" id="test_0"
value="Submit" /></div></td>
  </tr>
</table></form>
```

12.4.5 模板与主题

- 每种UI标签都有多种呈现模板（template）可供选择。例如，一种模板把<s:form>呈现为一个<form>元素和一个<table>元素，而另一种模板只把一个<s:form>标签呈现为一个表示元素，不增加<table>部分。
- 这些模板是用FreeMarker编写的，但使用这些模板不需要熟悉FreeMarker。

12.4.5 模板与主题

- 风格相近的模板被打包为一个主题（theme）。所谓主题就是为了让所有的UI标签能够产生同样的视觉效果而汇集到一起的一组模板。Struts 2目前提供了4种主题。
- `simple` 这个主题是模板最简单的，它把每个UI标签都封装在一个

元素中，使用这个主题时，每个UI标签都会被包裹在一个

元素中。例如，如果使用了这个主题，一个<s:form>标签将呈现为一个不带<table>元素的<form>元素，而一个<s:textfield>标签将呈现为一个不带任何修饰的<input>元素。

12.4.5 模板与主题

- 下面来看一下如何为UI标签设置一种主题。从前面例子中可以看到，如果没有为UI标签明确地指定一种主题，Struts 2就将使用xhtml主题里的模板。
- 为某个UI标签指定主题使用这个标签的theme属性。例如，下面这个<s:textfield>标签使用simple主题：

```
<s:textfield theme="simple"  
  name="userId" />
```

12.4.5 模板与主题

- 在表单里，如果没有给出一个UI标签的 `theme` 属性，它将使用所在表单的主题。例如，下面这些标签中，除最后一个 `<checkbox>` 标签使用 `simple` 主题外，其他的都使用 `css_xhtml` 主题。

```
<s:form theme="css_xhtml">  
  <s:checkbox name="daily" label="Daily news alert" />  
  <s:checkbox name="weekly" label="Weekly reports" />  
  <s:checkbox theme="simple" name="monthly"  
    label="Monthly reviews"  
    value="true" disabled="true" />  
  <s:submit value="提交" />  
</s:form>
```

本章内容

- 12.1 Struts 2框架概述
- 12.2 注册/登录系统
- 12.3 OGNL
- 12.4 Struts 2常用标签
- 12.5 用户输入校验
- 12.6 Struts 2的国际化
- 12.7 用Tiles实现页面布局

12.5 用户输入校验

- 一个健壮的Web应用程序必须确保用户输入是合法的。
- 例如，在把用户输入的信息存入数据库之前通常需要进行一些检查
 - 用户选择的口令达到一定长度（如不少于6个字符）
 - Email地址是合法的
 - 出生日期在合理的范围内等。
- 通常需要编写有关代码实现输入数据校验，在Struts 2中有多种方法实现用户输入校验。

12.5 用户输入校验

- 1.使用Struts 2校验框架。这种方法是Struts 2的基于XML的简单的校验方法，可以对用户输入数据自动校验。甚至可以使用相同的配置文件产生客户端脚本。
- 2.在Action类中执行校验。这是最强大和灵活的方法。Action中的校验可以访问业务逻辑和数据库等。但是，这种校验可能需要在多个Action中重复代码，并要求自己编写校验规则。而且，需要手动将这些条件映射到输入页面。

12.5 用户输入校验

- 3. **使用注解实现校验**。可以使用Java 5的注解功能定义校验规则，这种方法的好处是不用单独编写配置文件，所配置的内容和Action类放在一起，这样容易实现Action类中的内容和校验规则保持一致。
- 4. **客户端校验**。客户端校验通常是指通过浏览器支持的各种脚本来实现用户输入合法性的校验，这其中最经常使用的就是JavaScript。在Struts 2中可以通过有关标签产生客户端JavaScript校验代码。

12.5.1 使用Struts 2校验框架

- Struts 2的校验框架是内建校验程序，它大大简化了输入校验工作。使用该校验框架不需要编程，程序员只要在一个XML文件中对校验程序应该如何工作作出声明就行了。
- 需要声明的内容包括：哪些字段需要进行校验、在校验失败时把什么信息发送到浏览器。

12.6.1 使用Struts 2校验框架

- 在12.2节的用户注册和登录的例子中，对用户输入的数据没有提供任何校验功能。假设要求为用户输入定义下面的规则：
 - 必须提供用户名和口令字段值，口令不能少于6个字符。
 - 必须提供一个合法的Email地址。
 - 用户年龄必须在16到60之间。

12.5.1 使用Struts 2校验框架

- 校验程序配置工作的核心是编写校验程序配置文件，配置文件名格式为

<Action-Class-Name>-validation.xml

- 若要为RegisterAction动作类的属性进行校验，则配置文件名为RegisterAction-validation.xml，该文件应保存在与动作类相同的目录中。
- 程序12.19 RegisterAction-validation.xml

12.5.1 使用Struts 2校验框架

- 当输入校验失败后，Struts 2自动返回“input”的结果，因此需要在struts.xml文件中配置“input”的结果，如下代码所示。

```
<action name="Register"
  class="com.action.RegisterAction"
  method="register">
  <result name="success">/success.jsp</result>
  <result name="input">/register.jsp</result>
  <result name="error">/error.jsp</result>
</action>
```

12.5.1 使用Struts 2校验框架

用户注册

http://localhost:8080/basicstruts2/Register.action

注册一个新用户

用户名:

口令不能为空!

口令:

年龄:

邮箱地址不合法!

Email地址:

12.5.1 使用Struts 2校验框架

- Struts 2提供了大量的内建校验器，这些内建的校验器可满足大部分应用的校验需求，开发者只需使用这些校验器即可。如果应用有特别复杂的校验需求，而且该校验有很好的复用性，开发者可以开发自己的校验器。

12.5.1 使用Struts 2校验框架

- 在xwork-core-*VERSION*.jar中的com\opensymphony\xwork2\validator\validators路径下的default.xml文件中可以看到Struts 2的默认的校验器注册文件，里面定义了Struts 2所支持的全部校验器。表12.23列出了常用的内置校验器。

12.5.1 使用Struts 2校验框架

校验器名称	实现类	说 明
conversion	ConversionErrorFieldValidator	转换校验器。用于检查对指定字段进行类型转换时是否发生错误
date	DateRangeFieldValidator	日期范围校验器。用于检查指定字段的日期是否在给定的范围
double	DoubeRangeFieldValidator	浮点数范围校验器。用于检查指定字段的浮点数值是否在某个范围之内或之外
email	EmailValidator	邮件地址校验器。用于检查指定字段是否为一个合法的E-mail地址
expression	ExpressionValidator	表达式校验器。用于检查某个表达式的值是否是true
fieldexpression	FieldExpressionValidator	基于字段的表达式校验器。用于检查某个表达式的值是否是true

12.5.1 使用Struts 2校验框架

int	IntRangeFieldValidator	整数范围校验器。用于检查指定字段的整数值是否在某个范围之内
regex	RegexFieldValidator	正则表达式校验器。用于检查指定字段是否与给定的正则表达式相匹配
required	RequiredFieldValidator	必填校验器。用于检查指定的字段是否为空
requiredstring	RequiredStringValidator	必填字符串校验器。用于检查指定字符串非空且字符串的长度大于0
stringlength	StringLengthFieldValidator	字符串长度校验器。用于检查指定字符串的长度是否在某个范围之内
url	URLValidator	URL校验器。用于检查指定字段是否为合法的URL
visitor	VisitorFieldValidator	visitor校验器。用于实现对复合属性的校验

12.5.1 使用Struts 2校验框架

- 上面文件中定义的校验器使用的是**字段校验器语法**，在Struts 2中还可以使用**普通校验器**的方法，如下所示：

```
<validators>
```

```
  <validator type="email">
```

```
    <param  
name="fieldName">user.email</param>
```

```
    <message>邮件地址不合法! </message>
```

```
  </validator>
```

```
</validators>
```

12.5.1 使用Struts 2校验框架

- 在上面的数据校验中，校验失败的提示信息通过硬编码的方式写在配置文件中，这显然不利于程序的国际化。
- 在Struts 2中，数据的校验提示信息也可以实现国际化，这可通过为<message>元素提供key属性实现。例如，为user.username字段指定的校验规则可以使用key属性。

12.5.1 使用Struts 2校验框架

```
<field name="user.username">  
<field-validator type="requiredstring">  
    <param name="trim">true</param>  
<message key="username.required" />  
</field-validator>  
</field>
```

- 上述代码<message>元素指定了一个key属性，表明当user.username字段违反校验规则时，提示信息key为username.required的国际化消息。

12.5.2 使用客户端校验

- Struts 2的校验框架还可实现客户端校验，即产生客户端的JavaScript代码校验表单数据。
- 使用客户端校验非常简单，只要满足下面两个要求即可。（1）输入页面的表单元素使用Struts 2的标签生成。（2）在<s:form.../>元素增加validate="true"属性。
- 将JSP页面进行了上述修改后即可实现客户端校验，这里使用的校验配置文件仍然是RegisterAction-validator.xml。

12.5.2 使用客户端校验

- 当输入数据校验失败时显示的页面与服务器端校验效果相同。
- Struts 2将自动为JSP页面生成JavaScript校验代码并随响应数据一起发送到客户端。当在客户端打开页面时可以右击页面，从【查看源文件】中看到JavaScript校验代码。

12.5.2 使用客户端校验

- 注意，使用客户端校验并不支持所有的校验器。客户端校验仅支持下面的校验器：
required、requiredstring、stringlength、
regex、email、url、int和double校验器。

12.5.3 编程实现校验

- 前面介绍的校验方法是声明性的，即**先声明，后使用**。要校验的字段、使用的校验器和校验失败显示的信息都在XML配置文件中声明。
- 在某些场合，校验规则可能过于复杂，把它们写成一个声明性校验会非常复杂，因此Struts 2还提供了**通过编程方式实现校验**的功能。

12.5.3 编程实现校验

- Struts 2提供了`Validateable`接口，该接口只定义了`validate()`方法。
- 在动作类中可以实现该接口以提供编程校验功能。由于`ActionSupport`类已经实现了这个接口，所以如果动作类从`ActionSupport`类扩展而来，就可以直接覆盖`validate()`方法。

12.5.3 编程实现校验

- 下面例子说明如何通过覆盖validate()方法实现复杂的校验。
- 在注册程序中通常要求用户提供的口令具有一定的复杂度，例如要求口令至少包含一个数字、一个小写字母和一个大写字母才认为是一个强口令字，另外还可能要求口令字符串最少6个字符。

RegisterAction.java

本章内容

- 12.1 Struts 2框架概述
- 12.2 注册/登录系统
- 12.3 OGNL
- 12.4 Struts 2常用标签
- 12.5 用户输入校验
- 12.6 Struts 2的国际化
- 12.7 用Tiles实现页面布局

12.6 Struts 2的国际化

- 在程序设计领域，人们把能够在不改写有关代码的前提下，让开发出来的应用程序能够支持多种语言和数据格式的技术称为国际化技术。
- 在Web开发中要实现国际化技术，就是要求当应用程序运行时能够根据客户端请求所来自的国家/地区、语言的不同而显示不同的用户界面。

12.6.1 国际化

- 国际化通常简称i18n，来源是英文单词internationalization的首末字母i和n以及它们之间有18个字符。
- 引入国际化机制的目的在于提供自适应的、更友好的用户界面。
- Struts 2的国际化大致可分为页面的国际化、Action的国际化以及XML的国际化。
- 下面首先介绍属性文件，然后介绍Struts 2的国际化。

12.6.2 属性文件

- **属性文件（或资源文件）** 是用来保存多语言的字符串信息的文件。Java在实现软件的国际化时，采用了地区和语言两个因素来划分属性文件，也就是说，开发人员应该按照地区和语言来将字符串信息写到不同文件中。

1. 属性文件的格式

- 属性文件是纯文本文件。为了避免操作系统或编辑工具对不同语言的属性文件不支持，属性文件必须采用Unicode编码。
- 属性文件以行为单位，每行定义一个字符串资源，采用key=value的形式，key表示键的名称，value表示键的值。

2. 属性文件的命名

- 在Struts 2中，属性文件有不同级别，文件名也有不同的形式，但一般格式如下：

baseName_language_counrty.properties

所有的属性文件的扩展名都为.properties。

语言代码用两个小写字母表示，如en表示英语。国家代码用两个大写字母表示，如CN表示中国、US表示美国。

12.6.2 属性文件

- 下面是为LoginAction类指定的属性文件：

`LoginAction.properties`

`LoginAction_zh.properties`

`LoginAction_en_US.properties`

- 第一个文件没有使用语言代码和国家代码，它将使用默认语言和国家。第二个指定了语言代码，第三个指定了语言和国家代码。

12.6.3 属性文件的级别

- 在Struts 2中，对属性文件采取分级管理的方式。总体上说，属性文件可以分为以下三种类型：
 - 全局属性文件。
 - 包级别属性文件。
 - Action级别属性文件。
- 系统在查找属性文件时，查找顺序是从小范围到大范围，Action级的属性文件优先级最高，然后是包级别的属性文件，最后是全局属性文件。

1. 全局属性文件

- 全局属性文件可以被Struts 2应用的所有Action和JSP页面使用。全局属性文件只需在struts.xml或struts.properties文件中配置struts.custom.i18n.resources常量即可。
- 例如，设global.properties是全局属性文件，在struts.xml文件中定义如下：

```
<constant name="struts.custom.i18n.resources" value="global" />
```

1. 全局属性文件

- 该文件应该保存在WEB-INF/classes目录中，在Eclipse开发环境中应该保存在src目录中。
- 当Struts 2不能找到较低级别的属性文件时，将使用全局属性文件。

2. 包级别属性文件

- 包级别属性文件可以被一个包中的Action和JSP页面使用。包级别属性文件的`baseName`应该为`package`。
- 例如，`package.properties`和`package_zh_CN.properties`是两个包级别的属性文件。
- 包级别属性文件应该存放在包所在的目录中。

2. 包级别属性文件

- 假设在`com.action`包中建立一个名为`package.properties`的属性文件，在其中添加下面一行：

`greeting=欢迎来到Struts 2精彩世界！`

- 现在，任何由在`com.action`包中`Action`呈现的视图都可以使用`<s:text>`标签通过`greeting`属性名显示该键的值。例如，在JSP页面中可使用`<s:text>`标签显示`greeting`键的值：

`<h1><s:text name="greeting" /></h1>`

3. Action级别属性文件

- Action级别属性文件仅被当前Action类引用。在Struts 2应用程序中可以为每个Action类关联一个消息属性文件，属性文件名与Action类名相同，扩展名为`.properties`。
- 该属性文件必须存放在与Action类相同的包中。

12.6.4 Action的国际化

- 在Struts 2中可以在Action类中和JSP页面中使用属性文件，实现国际化。
- 在Action动作类中，只要其继承ActionSupport类就可以获得大部分的国际化的支持。ActionSupport类实现了com.opensymphony.xwork2.TextProvider接口，该接口负责提供对各种资源包和它们的底层文本消息的访问机制。

12.6.4 Action的国际化

- 在`TextProvider`接口中定义的
`getText(String key)`方法可以获得属性文件中某个键的值，`getText()`方法的参数为键名，结果为键值。在`Action`类的`execute()`方法中可以使用该方法。

12.6.4 Action的国际化

- 下面是getText()方法的常用格式：
- `public String getText(String key)`：返回与键相关的消息。如果找不到消息，返回空值null。
- `public String getText(String key, String defaultValue)`：返回与键相关的消息。如果找不到消息，返回defaultValue指定的默认值。

12.6.4 Action的国际化

- `public String getText(String key, String defaultValue, String[] args)`:
- 返回与键相关的消息，并使用给定的参数 `args` 的值填充占位符。如果找不到消息，返回 `defaultValue` 指定的默认值。该方法的第三个参数是 `String` 数组，也可以是字符串组成的 `List`。

12.6.5 JSP页面国际化

- 在JSP页面中可以使用`<s:text>`标签和`<s:i18n>`标签输出国际化字符串。
- 1. `<s:text>`标签
- 2. `<s:i18n>`标签

1. `<s:text>` 标签

- `<s:text>` 标签用来显示一条国际化消息，它是数据标签。它相当于从 `<s:property>` 标签中调用 `getText()` 方法。`<s:text>` 标签的属性如表 12.22 所示

属性名	数据类型	说 明
name	String	用来检索消息的键
var	String	用来引用被压入 Context Map 栈的值的变量名

1. <s:text>标签

- 例如，下面的<s:text>标签将输出与键label.hello相关联的消息。

```
<s:text  
  name="label.hello"></s:text>
```

- 若使用<s:property>标签，使用下面的方法输出与键label.hello相关联的值。

```
<s:property  
  value="%{getText('label.hello')}"  
/>
```

1. <s:text>标签

- 下面代码显示表单文本域，文本域的标题进行国际化。

```
<s:textfield name="name"  
    key="label.hello"/>
```

```
<s:textfield name="name"  
    label="%{getText('label.hello')}" /  
>
```


1. <s:text>标签

- 在使用<s:text>标签时如果给出了var属性，检索到的消息将被压入ValueStack栈的Stack Context子栈，而不是被输出。例如，下面代码将把与键greetings相关联的消息压入Stack Context子栈，并创建一个名为msg的变量来引用该消息：

```
<s:text name="greeting" var="msg"
/>
```

- 之后，我们就可以像下面这样使用<s:property>标签去访问这条消息了：

```
<s:property value="#msg" />
```

2. <s:i18n>标签

- <s:i18n>标签将加载一个自定义的资源包。该标签只有一个name属性，用来指定要加载的资源包的完全限定名。
- 下面代码使用<s:i18n>标签输出国际化字符串，messageResource为指定的资源文件名。例如：

```
<s:i18n name="messageResource">  
    <s:text  
        name="label.helloWorld"></s:text>  
</s:i18n>
```

12.6.6 Action属性文件应用

- Action级别属性文件是最常用的。
- 下面为12.2节的注册/登录的
`RegisterAction`动作类创建一个属性文件

`RegisterAction_en_US.properties`
文件存放在`com.action`包中。

12.6.6 Action属性文件应用

username=Username

password=Password

age=Age

email=Email Address

register=Register

thankyou=Thank you for registering
%{username}.

12.6.6 Action属性文件应用

- 下面是RegisterAction_zh_CN.properties属性文件也存放在com.action包中（该文件要使用native2ascii工具转换）。

username=用户名

password=口令

age=年龄

email=Email地址

register=注册

thankyou=谢谢注册%{userBean.username}.

label.hello=你好

12.6.6 Action属性文件应用

- 对于包含非西欧文字的属性文件还必须使用Java的native2ascii命令进行转换，该命令负责将非西欧文字转换成系统可以识别的文字。因此，必须对UserAction_zh_CN.properties属性文件进行转换。命令格式如下：

```
native2ascii source.properties  
destination.properties
```

12.6.6 Action属性文件应用

- 下面是由native2ascii工具生成的内容。

username=\u7528\u6237\u540d

password=\u53e3\u4ee4

age=\u5e74\u9f84

email=Email\u5730\u5740

register=\u6ce8\u518c

thankyou=\u8c22\u8c22\u6ce8\u518c%{user
Bean.username}.

label.hello=\u4f60\u597d

12.6.6 Action属性文件应用

- 为了JSP页面中使用国际化的属性值，修改register.jsp页面，将其中的<s:textfield>、<s:password>标签的label属性值和<s:submit>标签的属性值改为如下形式：

```
<s:textfield name="user.username"  
  label="%{getText('username')}}" />
```

```
<s:password name="user.password"  
  label="%{getText('password')}}" />
```

```
<s:textfield name="user.age" label="%{getText('age')}}"  
  />
```

```
<s:textfield name="user.email" label  
  ="%{getText('email')}}" />
```

```
<s:submit value="%{getText('register')}}" />
```


12.6.7 全局属性文件应用

- Struts 2提供了多种加载属性文件的方法。最简单、最常用的就是加载全局属性文件，这种方法是通过配置常量实现的。这只需在`struts.xml`（或`struts.properties`）文件中配置`struts.custom.i18n.resources`常量即可，即将该常量的值指定为`baseName`的值。

12.6.7 全局属性文件应用

- 假设系统需要加载的国际化属性文件的baseName为messageResource，则我们可以在struts.xml文件中指定下面一行：

```
<constant  
  name="struts.custom.i18n.resources"  
  value="messageResource" />
```

12.6.7 全局属性文件应用

- 或者在struts.properties文件中指定如下一行：

```
struts.custom.i18n.resources=messageResource
```

12.6.7 全局属性文件应用

- 下面创建两个资源文件。
 - messageResource_en_US.properties
 - messageResource_zh_CN.properties
- messageResource_en_US.properties的内容如下:

```
label.hello=hello,{0}  
label.helloWorld=Hello,World!  
userName=username  
userName.required=${getText('userName')} is required
```

12.6.7 全局属性文件应用

- messageResource_zh_CN.properties文件的内容如下:

label.hello=你好, {0}

label.helloWorld=你好, 世界!

userName=用户名

userName.required=\${getText('userName')} 不能为空

12.6.7 全局属性文件应用

- 在属性文件中可以包含参数占位符，它们用{0}、{1}等形式指定。这些参数占位符可以在执行ActionSupport类的getText()时为其传递参数，也可以在使用<s:text>标签时使用<s:param>子标签为其传递参数。

本章内容

- 12.1 Struts 2框架概述
- 12.2 注册/登录系统
- 12.3 OGNL
- 12.4 Struts 2常用标签
- 12.5 Struts 2的国际化
- 12.6 用户输入校验
- 12.7 用Tiles实现页面布局

12.7 用Tiles实现页面布局

- JSP提供了include指令（`<%@ include...`）包含静态文件和include动作（`<jsp:include ...>`）包含动态资源。
- 两种包含都存在着不足。一旦需要改变页面的布局，程序员将不得不去修改所有的页面。
- 使用Apache Tiles就可以避免上述不足。Tiles是一个模板框架，它用来简化Web应用的用户界面的开发。它允许定义页面片段来组装完整的页面。

12.7.1 安装所需的工具和库

- 在Struts 2应用程序中使用Tiles，首先应该从Struts 2的完整分发struts-2.3.8-all.zip文件中将下面JAR文件添加到WEB-INF/lib目录中。

- commons-beanutil-1.8.0.jar
- commons-collections-3.2.jar
- commons-digester-2.0.jar
- commons-logging-1.1.1.jar
- commons-logging-api-1.1.jar
- struts2-tiles-plugin-2.3.4.jar
- tiles-api-2.0.6.jar
- tiles-core-2.0.6.jar
- tiles-jsp-2.0.6.jar

12.7.2 在web.xml中配置Tiles

- 要使用Tiles，还应该在web.xml中注册一个监听器并指定Tiles定义文件tiles.xml的位置，代码如下。

```
<listener>
    <listener-class>
        org.apache.struts2.tiles.S
        trutsTilesListener
    </listener-class>
</listener>
```

12.7.2 在web.xml中配置Tiles

- 指定Tiles定义文件tiles.xml的位置

```
<context-param>
```

```
    <param-name>tilesDefinitions</param-  
    name>
```

```
    <param-value>/WEB-INF/tiles.xml</param-  
    value>
```

```
</context-param>
```

12.7.3 创建模板页面

- 模板页面是创建其他JSP页面的模板，在其中使用Tiles的<tiles:insertAttribute>标签作为占位符，这些占位符在定义文件中用具体的值或JSP页面替换。
- 模板页面baseLayout.jsp，代码如下。
- [程序12.20 baseLayout.jsp](#)

12.7.4 创建titles.xml定义文件

- 定义文件用来定义具体的视图页面。该文件首先用模板页面定义一个基本布局视图（baseLayout），然后定义两个视图继承该基本布局视图。
- 下面的定义文件名为tiles.xml，它应存放在/WEB-INF目录中。
- [程序12.22 tiles.xml](#)
- 每个<definition>元素定义一个逻辑视图名。

12.7.5 创建LoginAction类

- 为了展示Tiles，我们在LoginAction类中定义了登录验证authenticate()方法和退出登录logout()方法。
- 程序12.23 LoginAction.java

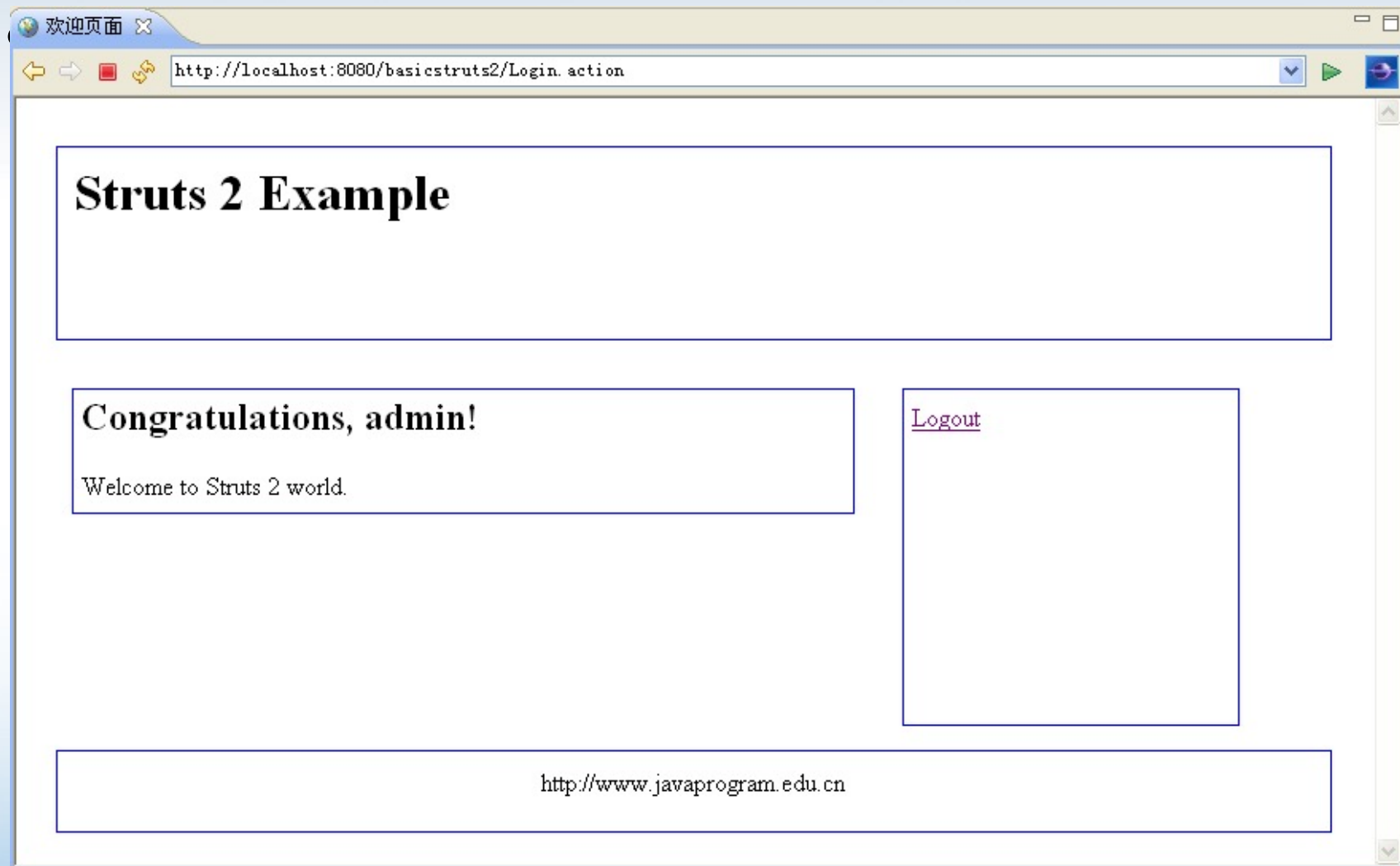
12.7.6 创建struts.xml文件

- 在struts.xml文件中的<package>元素中应添加调用tiles的标签处理请求，在动作的定义中将结果的type属性指定为“tiles”，结果指定为逻辑视图名。
- [程序12.24 struts.xml](#)

12.7.7 创建JSP页面

- 本应用程序中包含多个JSP页面。
- login.jsp登录页面如程序12.8所示，下面列出其他JSP页面代码。
- header.jsp页面构成页眉内容
- menu.jsp页面构成菜单
- footer.jsp页面构成页脚内容
- welcome.jsp页面是登录成功显示的页面

12.7.8 运行应用程序



12.8 小 结

- Struts 2框架实现了MVC体系结构，它通过提供一些类使用户很容易设计应用程序。它提供了一个核心控制器和一个Struts 2配置文件来管理所有的模型和视图。
- Struts 2提供了一组自定义标签，使用这些标签可以很容易在页面中输出数据、设计表单元素等。
- 使用Struts 2还可以方便地实现Web应用的表单数据校验、国际化以及页面布局等功能。