1 Struts2入门

本教程环境如下：

- JDK 1.8
- Maven 3.6
- IntelliJ IDEA 2019.3.1
- struts 2.5.22

1.1 Struts2简介

1.1.1 Struts2介绍

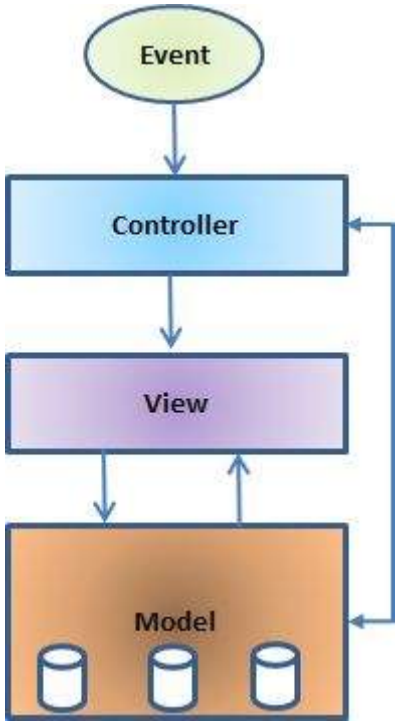
- Struts2是以WebWork设计思想为核心，在吸收Struts1部分优点的基础上设计出来的新一代的MVC框架
- Struts2是流行和成熟的基于MVC设计模式的Web应用程序框架。 Struts2不只是Struts1下一个版本，它是一个完全重写的Struts架构。
- WebWork框架开始以Struts框架为基础，其目标是提供一个加强和改进框架Struts来使web开发的开发人员更容易。

1.1.2 Struts MVC架构

模型（Model）—— 视图（View）—— 控制器（Controller），通常简称MVC，是一种开发web应用程序的软件设计模式。该软件设计模式由以下三部分组成:

- **模型**——属于软件设计模式的底层基础，主要负责数据维护。
- **视图**——这部分是负责向用户呈现全部或部分数据。
- **控制器**——通过软件代码控制模型和视图之间的交互。

MVC普及的原因在于它区分了应用程序的逻辑层和用户界面层，并支持开发关注点的分离。在MVC模式下，控制器接收了所有来自应用程序的请求后，调用模型去准备视图所需要的数据，然后视图使用由控制器提供的数据最终生成一个可视的响应。MVC的抽象概念可通过以下图形进行表述：



1.1.2.1 模型

模型主要负责管理应用程序的数据，它通过响应视图的请求和控制器的指令来更新自身的数据。

1.1.2.2 视图

通过控制器的指令触发所展现的一种特殊的数据格式。它们是基于像JSP、ASP、PHP之类模板系统的脚本，较易与AJAX技术进行整合。

1.1.2.3 控制器

控制器负责响应用户输入并执行数据模型对象的交互。控制器在接收、确认输入后执行修改数据模型状态的业务操作。

1.1.3 Struts2优点

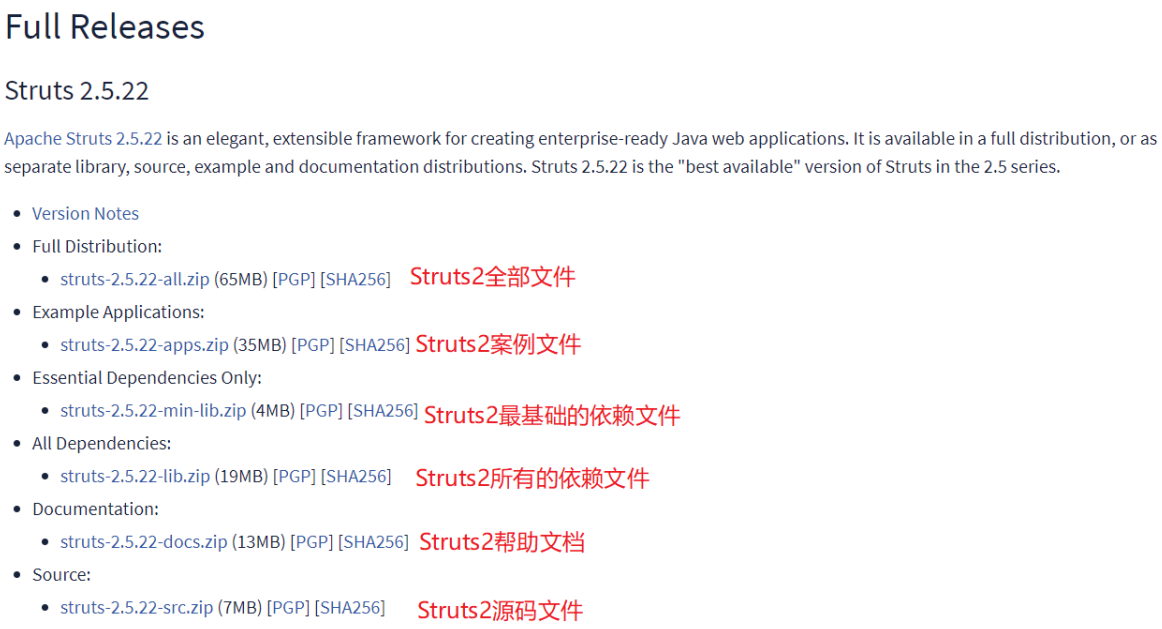
1. 实现MVC模式，结构清晰,使开发者只关注业务逻辑的实现.
2. 有丰富的tag可以用 ,Struts的标签库(Taglib)，如能灵活动用，则能大大提高开发效率
3. 页面导航，使系统的脉络更加清晰。通过一个配置文件，即可把握整个系统各部分之间的联系，这对于后期的维护有着莫大的好处。尤其是当另一批开发者接手这个项目时，

这种优势体现得更加明显。

4. 提供Exception处理机制 .
5. 数据库连接池管理
6. 支持 I18N （国际化）

1.1.4 获取Struts2

> 官网地址：https://struts.apache.org/index.html
>
> 下载地址：http://struts.apache.org/download.cgi



Full Releases

Struts 2.5.22

Apache Struts 2.5.22 is an elegant, extensible framework for creating enterprise-ready Java web applications. It is available in a full distribution, or as separate library, source, example and documentation distributions. Struts 2.5.22 is the "best available" version of Struts in the 2.5 series.

- Version Notes
- Full Distribution:
  - struts-2.5.22-all.zip (65MB) [PGP] [SHA256] Struts2全部文件
- Example Applications:
  - struts-2.5.22-apps.zip (35MB) [PGP] [SHA256] Struts2案例文件
- Essential Dependencies Only:
  - struts-2.5.22-min-lib.zip (4MB) [PGP] [SHA256] Struts2最基础的依赖文件
- All Dependencies:
  - struts-2.5.22-lib.zip (19MB) [PGP] [SHA256] Struts2所有的依赖文件
- Documentation:
  - struts-2.5.22-docs.zip (13MB) [PGP] [SHA256] Struts2帮助文档
- Source:
  - struts-2.5.22-src.zip (7MB) [PGP] [SHA256] Struts2源码文件

1.2 入门案例

1. 添加struts2相关依赖
2. 修改web.xml,加载struts2配置
3. 编写index.jsp页面
4. 编写Action控制器
5. 编写struts.xml文件

1.2.1 添加struts2相关依赖

> 在 pom.xml 文件加入struts2的依赖代码

```xml
<!-- struts2核心依赖 -->
<dependency>
  <groupId>org.apache.struts</groupId>
  <artifactId>struts2-core</artifactId>
  <version>2.5.22</version>
</dependency>
```

1.2.2 修改web.xml,加载struts2配置

```xml
<!-- struts2核心配置 -->
<filter>
    <!-- 过滤器名称，自定义，命名为struts2 -->
    <filter-name>struts2</filter-name>
    <!-- 过滤器核心类 -->
    <filter-
class>org.apache.struts2.dispatcher.filter.StrutsPrepareAndExecuteFilter</filter-class>
</filter>
<filter-mapping>
    <!-- 过滤器名称，自定义，命名为struts2 -->
    <filter-name>struts2</filter-name>
    <!-- 过滤范围 -->
    <url-pattern>/*</url-pattern>
</filter-mapping>
```

1.2.3 编写index.jsp页面

```jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户登录</title>
</head>
<body>

<form action="hello.action" method="post">
    <div>
        <label>用户名: </label>
        <input type="text" name="userName">
    </div>
    <div>
        <input type="submit" value="登录">
    </div>
</form>

</body>
</html>
```

1.2.4 编写Action控制器

1. 控制器需要实现Action接口，该Action接口位于 `com.opensymphony.xwork2` 包下
2. 获取页面的数据可以在Action中定义成员变量，必须与input标签的name属性值一致，并且必须提供get、set方法
3. execute()方法需要返回逻辑字符串(逻辑字符串自定义)

```java
package com.bdqn.web;

import com.opensymphony.xwork2.Action;

/**
 * 控制器
 */
public class HelloAction implements Action {

    //定义变量，接收页面的用户名userName
    //要求：属性名必须与input标签的name属性值一致，并且必须提供get、set方法
    private String userName;


    @Override
    public String execute() throws Exception {
        System.out.println("userName:"+userName);
        //返回逻辑字符串（自定义）
        return "success";
    }


    public String getUserName() {
        return userName;
    }

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

1.2.5 编写struts.xml配置文件

1. struts2配置文件名定义成 `struts.xml`
2. 该配置文件需要定义在resources目录下

```xml
<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE struts PUBLIC
        "-//Apache Software Foundation//DTD Struts Configuration 2.5//EN"
```

```
        "http://struts.apache.org/dtds/struts-2.5.dtd">

<struts>

    <!--
        package标签：该标签配置控制器的包结构
            name属性：自定义，必须保证唯一,与编码没有任何关系，用于区分package
            extends属性：继承某个package,其中struts-default是struts2框架的核心
            namespace属性：命名空间，默认为/(根目录)
     -->
    <package name="default" extends="struts-default" namespace="/">

        <!--
            action标签：配置访问请求
                name属性：访问路径
                class属性：访问该路径要执行的控制器
        -->
        <action name="hello" class="com.bdqn.web.HelloAction">
            <!--
                result标签：配置返回的页面
                    name属性：属性值可以自定义，必须与控制器返回的逻辑字符串一致
            -->
            <result name="success">/show.jsp</result>
        </action>

    </package>

</struts>
```

1.2.6 输出结果(show.jsp)

> 在JSP页面中输出结果的方式有两种，分别是:
>
> 1. EL表达式，例如：${成员变量名称}
> 2. struts2标签，例如：`<s:property value='成员变量名称'/>`
>
> 注意：
>
> 1. struts2标签不能使用其他的表达式，如：在struts2标签中使用EL表达式,会报错
> 2. 使用Struts2标签输出结果，必须在页面中引入相应的标签库 `<%@ taglib prefix="s" uri="/struts-tags" %>`

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>显示</title>
</head>
<body>

    <h1>用户名：${userName}</h1>

```

```
    <h1>用户名: <s:property value="userName"/></h1>

</body>
</html>
```

1.3 使用Struts2实现用户登录

1.3.1 开发思路

1. 编写login.jsp页面(用于登录)
2. 编写登录的控制器（LoginAction）
    1. 获取登录表单的用户名和密码(成员变量、提供get和set方法)
    2. 重写execute()方法
3. 编写struts.xml配置文件

1.3.2 编写登录页面(login.jsp)

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户登录</title>
</head>
<body>

<form action="login.action" method="post">
    <div>
        <label>用户名: </label>
        <input type="text" name="userName">
    </div>
    <div>
        <label>密码: </label>
        <input type="password" name="password">
    </div>
    <div>
        <input type="submit" value="登录">
    </div>
</form>

</body>
</html>
```

1.3.3 编写LoginAction控制器

```java
package com.bdqn.web;

import com.opensymphony.xwork2.Action;

public class LoginAction implements Action {

    //获取用户名
    private String userName;//与input标签的name属性值保持一致，提供get、set方法
    //获取密码
    private String password;//与input标签的name属性值保持一致，提供get、set方法

    @Override
    public String execute() throws Exception {
        //比较用户名和密码是否一致
        if(userName.equals("admin") && password.equals("123456")){
            return "success";//成功
        }
        return "error";//失败
    }


    //以下是get、set方法
    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public String getPassword() {
        return password;
    }

    public void setPassword(String password) {
        this.password = password;
    }
}
```

1.3.4 编写struts.xml配置文件

```xml
<!-- 用户登录请求 -->
<action name="login" class="com.bdqn.web.LoginAction">
    <!-- 成功 -->
    <result name="success">/success.jsp</result>
    <!-- 失败 -->
    <result name="error">/login.jsp</result>
</action>
```

1.4 Action接口

Actions是Struts2框架的核心，因为它们适用于任何MVC（Model View Controller）框架。每个URL映射到特定的action，其提供处理来自用户的请求所需的处理逻辑。

但action还有另外两个重要的功能。 首先，action在将数据从请求传递到视图（无论是JSP还是其他类型的结果）方面起着重要作用。 第二，action必须协助框架确定哪个结果应该呈现在响应请求的视图中。

Struts2中actions的唯一要求是必须有一个无参数方法返回String或Result对象，并且必须是POJO(类)。如果没有指定no-argument无参方法，则默认是使用execute()方法。其Action接口源码如下：

```
public interface Action {
   public static final String SUCCESS = "success";//成功
   public static final String NONE = "none";//无
   public static final String ERROR = "error";//错误
   public static final String INPUT = "input";//未输入数据
   public static final String LOGIN = "login";//登录
   public String execute() throws Exception;
}
```

1.5 Struts 2访问Servlet API

在struts2中使用Servlet提供的方法的方式有两种，分别是：

1. 与Servlet API解耦的访问方式
2. 与Servlet API耦合的访问方式

> 解耦：组件框架之间相互独立，能够独自正常运行，不进行依赖操作
>
> 耦合：组件框架之间相互依赖，缺少其中一个组件则可能无法正常运行

1.5.1 与Servlet API解耦的访问方式

1. 对Servlet API进行封装

    提供了三个Map对象访问request、session、application作用域

2. 通过 `ActionContext` 类获取这三个Map对象

    1. `Object get("request")` ：获取request对象
    2. `Map getSession()` ：获取Session对象
    3. `Map getApplication()` ：获取Application对象

```java
@Override
public String execute() throws Exception {
    //比较用户名和密码是否一致
    if(userName.equals("admin") && password.equals("123456")){
        //使用解耦的方式获取Session，将当前登录用户的信息保存到session
        Map<String, Object> session =  ActionContext.getContext().getSession();
        //将当前用户信息保存到session中
        session.put("loginUser",userName);
        //等同于session.setAttribute("loginUser",userName);
        //return "success";//成功
        return SUCCESS;//使用Action接口中的常量
    }
    return LOGIN;
}
```

1.5.2 与Servlet API耦合的访问方式

1. 通过 `ServletActionContext` 类获取Servlet API对象

    1. `ServletContext getServletContext()` ：获取Application
    2. `HttpServletResponse getResponse()` ：获取response对象
    3. `HttpServletRequest getRequest()` ：获取request对象

2. 通过 `request.getSession()` 获取 `session` 对象

    通过 `xxx.setAttribute()` 和 `xxx.getAttribute()` 方法在不同的页面或Action中传递数据

```java
@Override
public String execute() throws Exception {
    //比较用户名和密码是否一致
    if(userName.equals("admin") && password.equals("123456")){
        //使用耦合的方式获取Session，此时必须依赖Servlet
        HttpSession session = ServletActionContext.getRequest().getSession();
        //将当前用户信息保存到session中
        session.setAttribute("loginUser",userName);
        //return "success";//成功
        return SUCCESS;//使用Action接口中的常量
    }
    //return "error";//失败
    return LOGIN;
}
```

注意：使用耦合的方式访问Servlet API，必须加入Servlet的依赖

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>4.0.1</version>
  <scope>provided</scope>
</dependency>
```

1.6 数据校验

1.6.1 概述

- Struts 2提供了数据验证机制
- 继承ActionSupport类来完成Action开发
- ActionSupport类不仅对Action接口进行简单实现，同时增加了验证、本地化等支持

1.6.2 数据校验步骤

第1步：控制器Action继承 `ActionSupport` 类

第2步：控制器Action重写 `validate()` 方法

第3步：在页面中使用 `<s:fielderror>` 标签显示验证信息

第4步：在 `struts.xml` 文件的action标签中配置 `input` 返回结果

1.6.3 实现数据校验

1.6.3.1 控制器Action继承ActionSupport类

```
package com.bdqn.web;

import com.opensymphony.xwork2.ActionSupport;

//继承ActionSupport
public class LoginAction extends ActionSupport {

}
```

1.6.3.2 控制器Action重写validate()方法

> StringUtils.isEmpty() 方法是 org.apache.commons.lang3 包下的方法

```
@Override
public void validate() {
    //判断用户名是否为空
    if(StringUtils.isEmpty(userName)){
        //参数1：key自定义
        //参数2：验证提示信息
        addFieldError("userName","用户名不能为空");
    }

    //密码
    if(password==null || password.length()==0){
        addFieldError("password","密码不能为空");
    }
}
```

1.6.3.3 在页面中使用<s:fielderror>标签显示验证信息

注意<s:fielderror>标签如果不指定fieldName的属性值，则默认显示所有验证信息,其中fieldName属性是validate()方法中的key

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户登录</title>
</head>
<body>

<s:fielderror/>

<form action="login.action" method="post">
    <div>
        <label>用户名：</label>
        <input type="text" name="userName">
        <font color="red"><s:fielderror fieldName="userName"/></font>
    </div>
    <div>
        <label>密码：</label>
        <input type="password" name="password">
        <font color="red"><s:fielderror fieldName="password"/></font>
    </div>
    <div>
        <input type="submit" value="登录">
    </div>
</form>
```

```
26
27 </body>
28 </html>
```

1.6.3.4 在struts.xml文件的action标签中配置input返回结果

```
1 <!-- 用户登录请求 -->
2 <action name="login" class="com.bdqn.web.LoginAction">
3     <!-- 成功 -->
4     <result name="success">/success.jsp</result>
5     <!-- 失败 -->
6     <result name="login">/login.jsp</result>
7     <!-- input -->
8     <result name="input">/login.jsp</result>
9 </action>
```

1.7 Struts2标签

1.7.1 概述

struts2有丰富的tag(标签)可以使用 ,Struts2的标签库(Taglib)，如能灵活动用，则能大大提高开发效率

struts2标签分为UI标签和通用标签，使用前需要引入struts2的标签库。

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

1.7.2 UI标签

UI标签分为：表单标签，非表单标签和Ajax标签

1.7.2.1 表单标签

标　签	说　明
<s:form>...</s:form>	表单标签
<s:textfield>...</s: textfield>	文本输入框
<s:password>...</s: password>	密码输入框
<s:textarea>...</s: textarea>	文本域输入框
<s:radio>...</s: radio>	单选按钮
<s:checkbox>...</s: checkbox>	多选框
<s:submit/>	提交标签
<s:reset/>	重置标签
<s:hidden/>	隐藏域标签

```
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户注册</title>
</head>
<body>

<s:form method="POST" action="#">
    <div>
        <label>用户名: </label>
        <s:textfield id="userName" name="userName"/>
    </div>
    <div>
        <label>密码: </label>
        <s:password name="password"/>
    </div>
    <div>
        <label>电话: </label>
        <s:textfield name="phone"/>
    </div>
    <div>
        <s:submit value="注册"/>
    </div>
</s:form>


</body>
</html>
```

注意：使用struts2标签，会导致样式错乱，需要在struts.xml配置文件指定配置，代码如下：

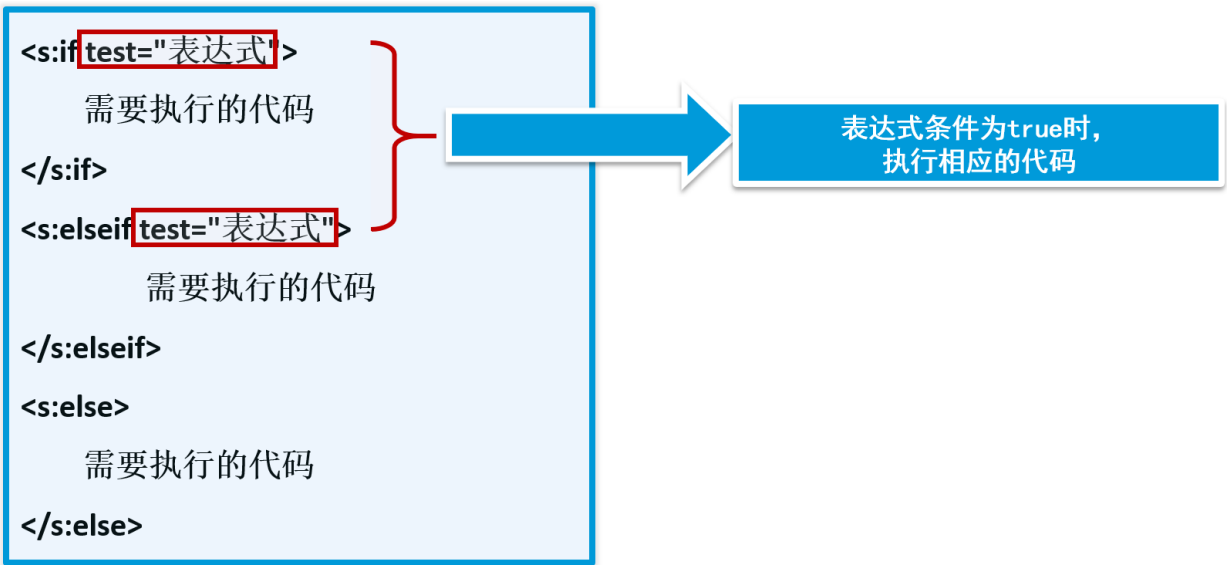
```
<constant name="struts.ui.theme" value="simple"></constant>
```

1.7.3 通用标签

通用标签分为 条件标签 和 迭代标签

名称	标　签	说　明
条件标签	<s:if>.....</s:if>	根据表达式的值， 判断将要执行的内容
	<s:elseif>.....</s:elseif>	
	<s:else>.....</s:else>	
迭代	<s:iterator>.....</s:iterator>	用于遍历集合

1.7.3.1 条件标签



1.7.3.2 迭代标签

```
<s:iterator value="集合对象" status="status" id="name">
        读取集合对象并输出显示
</s:iterator>
```

- value属性：需要进行遍历的集合对象
- status属性：当前迭代元素的IteratorStatus实例
- id属性:当前迭代元素的id，可直接访问元素，该参数可选

1.8 综合案例——查询用户列表

1.8.1 编写User实体类

```java
package com.bdqn.entity;

public class User {
    private Integer id;
    private String userName;
    private Integer sex;//1-男 , 2-女
    private String phone;

    public Integer getId() {
        return id;
    }

    public void setId(Integer id) {
        this.id = id;
    }

    public String getUserName() {
        return userName;
    }

    public void setUserName(String userName) {
        this.userName = userName;
    }

    public Integer getSex() {
        return sex;
    }

    public void setSex(Integer sex) {
        this.sex = sex;
    }

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public User(){

    }

    public User(Integer id, String userName, Integer sex, String phone) {
        this.id = id;
        this.userName = userName;
        this.sex = sex;
        this.phone = phone;
    }
}
```

1.8.2 编写UserAction控制器

注意：该控制器继承ActionSupport，自定义userList()方法

```java
package com.bdqn.web;

import com.bdqn.entity.User;
import com.opensymphony.xwork2.ActionSupport;

import java.util.ArrayList;
import java.util.List;

public class UserAction extends ActionSupport {

    //定义成员变量，保存用户列表
    private List<User> userList;//值栈，类似request.getParameter()和xxx.setAttribute()的组合

    public String userList() throws Exception {
        userList = new ArrayList<User>();
        userList.add(new User(1,"张三",1,"13212345678"));
        userList.add(new User(2,"李四",1,"13212345677"));
        userList.add(new User(3,"王五",1,"13212345676"));
        userList.add(new User(4,"王丽",2,"13212345675"));
        userList.add(new User(5,"小红",2,"13212345674"));
        //返回逻辑字符串
        return SUCCESS;
    }



    public List<User> getUserList() {
        return userList;
    }

    public void setUserList(List<User> userList) {
        this.userList = userList;
    }
}
```

1.8.3 编写struts.xml文件

注意：由于控制器没有重写execute()方法，而是自定义了userList()方法，所以需要在<action>标签通过method属性指定方法名称

```xml
<!-- method属性指定控制器的方法名称 -->
<action name="findUserList" class="com.bdqn.web.UserAction" method="userList">
    <!-- name属性默认值是success，可以不写 -->
    <result name="success">/userlist.jsp</result>
</action>
```

1.8.4 userlist.jsp

```jsp
<%@ taglib prefix="s" uri="/struts-tags" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>用户列表</title>
</head>
<body>

<table border="1" cellspacing="0" width="1000" align="center">
    <tr>
        <th>编号</th>
        <th>姓名</th>
        <th>性别</th>
        <th>电话</th>
    </tr>

    <%-- 循环标签，value属性：循环的集合或数组 --%>
    <s:iterator value="userList">
        <tr>
            <td><s:property value="id"/></td>
            <td><s:property value="userName"/></td>
            <td>
                <s:if test="sex==1">男</s:if>
                <s:else>女</s:else>
            </td>
            <td><s:property value="phone"/></td>
        </tr>
    </s:iterator>

</table>

</body>
</html>
```