实验三 **Struts2** 的控制器组件 **Action**——登录用户的功能扩

展

一、基础实验——**Action** 的自定义方法

# （一）实验目的

1. 掌握 Struts2 的 Action 类中自定义方法的使用；
2. 掌握 Struts2 中 Action 类的不同调用方式和相应的配置方法；
3. 掌握 Action 的实例化情况，理解 Action 与 Servlet 在实例化上的区别；
4. 理解 JSP 文件中获取 Action 属性的过程；
5. 了解 Struts2 支持的 Action 处理结束后的结果类型。

# （二）基本知识与原理

1. Action 类中的默认方法名是 execute()方法，可以被自动调用；
2. 在 Action 中也允许定义其它方法名，可以同时定义多个方法，分别处理不同的逻辑；
3. 当 Action 中使用了自定义方法，则该 Action 就需要特定的配置，一般有四种调用方式：
4. 在 struts.xml 文件中通过 method 属性指定方法名；
5. 使用动态方法调用方式（DMI）；
6. 使用提交按钮的 method 属性；
7. 使用通配符配置 Action；
8. Action 类是多实例的，Action 类的属性是线程安全的；
9. 在 JSP 页面中，可以通过 Struts2 标签调用 Action 中对应的 getter 方法，从而输出 Action 的属性值；
10. 当一个 Action 处理用户请求结束后，返回一个字符串作为逻辑视图名，再通过 struts.xml 文件中的配置将逻辑视图名与物理视图资源关联起来；Struts2 默认提供了一系列的结果类型（struts-default.xml 配置文件的 result-types 标签里列出了所支持的结果类型），结果类型决定了 Action 处理结束后，将调用哪种视图资源来呈现处理结果。

# （三）实验内容及步骤

1. 在 Eclipse 中新建 Web 工程 struts-prj2，并将 Struts2 中的 8 个核心包添加到工程中；
2. 在 struts-prj2 中新建 login.jsp 页面，作为用户登录的视图；新建 loginFail.jsp 页面，作为登录失败的视图；新建 loginSuccess.jsp 页面，作为登录成功的视图（可重用“实验二 Struts 基础应用”中基础实验里的页面代码）；
3. 在 struts-prj2 中新建 register.jsp 页面，作为用户注册的视图；新建 regFail.jsp 页面，作为注册失败的视图；新建 regSuccess.jsp 页面，作为注册成功的视图（可重用“实验二 Struts 基础应用”中提高实验里的页面代码）；
4. 在 struts-prj2 中新建 cn.edu.zjut.bean 包，并在其中创建 UserBean.java，用于记录用户信息（可重用“实验二 Struts 基础应用”中提高实验里的

UserBean.java 代码）；

1. 在 struts-prj2 中新建 cn.edu.zjut.service 包，并在其中创建 UserService.java，用于实现登录逻辑和注册逻辑（可重用“实验二 Struts 基础应用”中提高实

验里的 UserService.java 代码）；

1. 在 struts-prj2 中新建 cn.edu.zjut.action 包，并在其中创建 UserAction.java，定义 login()方法和 register()方法，参照“实验二 Struts 基础应用”写入代码，

分别用于调用登录逻辑和注册逻辑（代码片段如下）；

|  |
| --- |
| package cn.edu.zjut.action;  ……  public class UserAction {  ……  public String **login()** {  UserService userServ = new UserService(); if (userServ.login(loginUser)) { return "success"; }  return "fail";  }  public String **register()** {  UserService userServ = new UserService(); if (userServ.register(loginUser)) { return "registersuccess"; }  return "registerfail";  }  } |

1. 在工程 struts-prj2 的 src 目录中创建 struts.xml 文件，用于配置 Action 并设置页面导航，通过 action 标签中 method 属性指定方法名（代码片段如下）；

|  |
| --- |
| <struts>  <package name="strutsBean" extends="struts-default" namespace="/">  <action name="login" class="cn.edu.zjut.action.UserAction" **method="login"**>  <result name="success">/loginSuccess.jsp</result>  <result name="fail">/loginFail.jsp</result>  </action>  <action name="register" class="cn.edu.zjut.action.UserAction" **method="register"**>  <result name="registersuccess">/regSuccess.jsp</result>  <result name="registerfail">/regFail.jsp</result> </action>  </package>  </struts> |

1. 编辑 Web 应用的 web.xml 文件，增加 Struts2 核心 Filter 的配置；
2. 将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 与 register.jsp 页面，并记录运行结果；
3. 查找相关资料，尝试使用 Action 自定义方法的其它三种调用和配置方式：动态方法调用方式（DMI）、提交按钮的 method 属性、通配符配置 Action，并记录关键配置和运行结果；
4. 修改 UserAction.java，增加 UserAction 类的构造方法 UserAction()，增加 count 属性，用于测试 Action 的实例化情况（代码片段如下）；

|  |
| --- |
| public class UserAction { private Integer count=0; **public UserAction(){**  **System.out.println("**创建了一个**UserAction**类对象。**");**  **}** public Integer getCount() { return count;  } public String login() { **count++; //Action**实例化情况测试  UserService userServ = new UserService(); if (userServ.login(loginUser)) { return "success";  } return "fail";  }  ……  } |

1. 修改 loginSuccess.jsp，在页面中使用<s:property>标签输出 Action 中的 count 值；
2. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，并刷新多次，记录运行结果；
3. 修改 struts.xml 文件，将 UserAction 的页面导航设置为 redirect 结果类型（代码片段如下）；

|  |
| --- |
| <struts>  <package name="strutsBean" extends="struts-default" namespace="/">  <action name="login" class="cn.edu.zjut.action.UserAction" method="login">  <result name="success" **type="redirect"**>  /loginSuccess.jsp</result>  <result name="fail">/loginFail.jsp</result> </action>  </package>  </struts> |

1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察登录成功后 loginSuccess.jsp 页面的输出，并记录下来。

# （四）实验要求

1、填写并上交实验报告，报告中应包括：

1. 运行结果截图；
2. 查找相关资料，根据实验过程，总结 Action 自定义方法的四种调用和配置方式；

在struts.xml文件中通过method属性指定方法名：写多个action，每个action中的class相同，通过不同的method调用同一个action类中不同的方法

使用动态方法调用方式（DMI）：action一个，不在action配置中区分方法。而在表单提交的action中通过！添加要调用的方法

使用提交按钮的method属性：action一个，不在action配置中区分方法。而在表单中submit控件的method方法添加要调用的方法

使用通配符配置Action：通过通配符代替特定标识，action配置中的method由通配符决定也就是由传入的特定标识决定

1. 根据实验过程，记录实验步骤 11、12 中修改后的关键代码，及相应的运行结果或报错信息，分析 Action 的实例化情况，将 Action 与 Servlet 在实例化情况上进行对比，并记录下来；

Action多次实例化，所以每次count都是1

Servlet在容器加载servlet的时候实例化，与请求无关

1. 根据实验步骤 11、12，查找相关资料，分析 JSP 文件中获取 Action 属性的主要过程，并记录下来；

表单提交请求，根据struts.xml找到action类并实例化处理请求，再根据struts.xml转发请求到相应的jsp页面，jsp通过<property>标签获得属性值

1. 根据实验步骤 11-15，观察两次 loginSuccess.jsp 页面输出上的区别，分析原因并记录下来；

重定向不像请求转发，它跳转后已经不是提交表单的那个了，自然没有count这个属性了

1. 解压缩 Struts2 的核心包 struts2-core-2.3.15.1.jar，找到 struts-default.xml 配置文件，在其中的 result-types 标签里列出了 Struts2 所支持的结果类型，查找相关资料，总结这些结果类型的作用和特点，并记录下来；

<package name="struts-default" abstract="true" strict-method-invocation="true">  
 <result-types>  
 <result-type name="chain" class="com.opensymphony.xwork2.ActionChainResult"/>  
 <result-type name="dispatcher" class="org.apache.struts2.result.ServletDispatcherResult" default="true"/>  
 <result-type name="freemarker" class="org.apache.struts2.views.freemarker.FreemarkerResult"/>  
 <result-type name="httpheader" class="org.apache.struts2.result.HttpHeaderResult"/>  
 <result-type name="redirect" class="org.apache.struts2.result.ServletRedirectResult"/>  
 <result-type name="redirectAction" class="org.apache.struts2.result.ServletActionRedirectResult"/>  
 <result-type name="stream" class="org.apache.struts2.result.StreamResult"/>  
 <result-type name="velocity" class="org.apache.struts2.result.VelocityResult"/>  
 <result-type name="xslt" class="org.apache.struts2.views.xslt.XSLTResult"/>  
 <result-type name="plainText" class="org.apache.struts2.result.PlainTextResult" />  
 <result-type name="postback" class="org.apache.struts2.result.PostbackResult" />  
 </result-types>

|  |  |
| --- | --- |
| ****名字**** | ****说明**** |
| Chain Result | 用来处理Action链 |
| Dispatcher Result | 用来转向页面，通常处理JSP |
| FreeMarker Result | 处理FreeMarker模板 |
| HttpHeader Result | 用来控制特殊的Http行为 |
| Redirect Result | 重定向到一个URL |
| Redirect Action Result | 重定向到一个Action |
| Stream Result | 向浏览器发送InputSream对象，通常用来处理文件下载 |
| Velocity Result | 处理Velocity模板 |
| XLS Result | 处理XML/XLST模板 |
| PlainText Result | 显示原始文件内容，例如文件源代码 |
| S2PLUGINS:Tiles Result | 结合Tile使用 |

1. 碰到的问题及解决方案或思考；
2. 实验收获及总结。

2、上交程序源代码，代码中应有相关注释。

二、提高实验——**ActionSupport** 与输入校验

# （一）实验目的

1. 了解 Action 接口的作用，理解 ActionSupport 类的作用；
2. 掌握在 Struts2 中使用校验器或手工编码的方式，对请求参数进行数据校验的方法，掌握在 JSP 页面中显示错误信息和提示信息的方法；
3. 掌握在 Action 中使用国际化资源文件的方法；
4. 掌握 Struts2 内置类型转换器的作用和使用方法。

# （二）基本知识与原理

1. 为了让用户开发的 Action 类更规范，Struts2 提供了一个 Action 接口，该接口定义了 Struts2 的 Action 处理类应该实现的规范；
2. Struts2 还为 Action 接口提供了一个实现类：ActionSupport，该类提供了若干默认方法，包括：默认的处理用户请求的方法（excute()方法）、数据校验的方法、添加校验错误信息的方法、获取国际化信息的方法等，部分重要方法列表如下：

表 3-1 ActionSupport 类的部分重要方法

|  |  |
| --- | --- |
| 方法名 | 说明 |
| public String execute() | 默认的处理用户请求的方法，直接返回 SUCCESS 字符串。 |
| public void validate() | 空的输入校验方法，常被 Action 类覆盖，实现对输入参数的校验。 |
| public void addActionError(String anErrorMessage) | 将 Action 级别的错误信息添加到 Action 中。 |
| public void addActionMessage(String aMessage) | 将 Action 级别的消息添加到 Action 中。 |
| public void addFieldError(String fieldName, String errorMessage) | 将域级错误信息添加到特定的域中。 |
| public String getText(String aTextName) | 从国际化资源文件中获取属性值，其中参数是属性文件的 key 值 |

1. Struts2 框架提供了校验器和手工编码两种方式对请求参数进行数据校验，当 Action 类继承了 ActionSupport 类，就可以通过定义名为“<ActionClassName><ActionAliasName>-validation.xml”的校验规则文件的方法进行校验器校验，也可以通过重写 ActionSupport 类的 validate()方法或 validateXxx()方法进行手动校验；
2. 在 JSP 页面中使用 Struts2 标签生成的表单，能将域级别的错误信息将自动显示到表单元素处；
3. 在 JSP 页面中使用 fielderror 标签，可以集中显示所有的域级错误信息；使用 actionerror 标签，可以显示所有的 Action 级别错误信息；使用 actionmessage 标签，可以显示 Action 消息；
4. Struts2 框架中提供了部分内置的类型转换器，可以将请求参数的 String 类型转换成基本数据类型及对应的包装器类型、日期类型、数组类型、集合类型等，当 Action 类继承了 ActionSupport 类，则内置的类型转换器将默认生效，可以直接使用；
5. 如需修改默认的类型转换校验信息，则要在 Action 类的包中声明名为“Action 类名.properties”的局部属性文件；
6. Struts2 框架同时支持自定义类型转换器，将请求参数转换成任意一种类型。

# （三）实验内容及步骤

1. 在struts-prj2中修改UserAction类，使其继承ActionSupport类，并在UserAction 类中覆盖 ActionSupport 类的 validate()方法，用于对用户登录的请求参数 account 和 password 进行校验：若用户名或密码为空，则使用 addFieldError

（域级）添加错误信息（代码片段如下）；

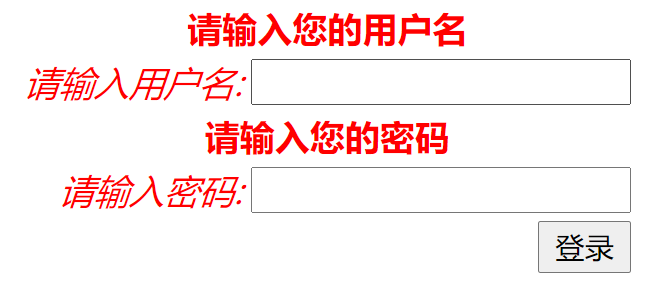
|  |
| --- |
| **import com.opensymphony.xwork2.ActionSupport;** …… public class UserAction **extends ActionSupport** {  ……  **public void validate()** {  String account = loginUser.getAccount(); String pwd = loginUser.getPassword(); if (account == null || account.equals("")) { **this.addFieldError("loginUser.account", "**请输入您的用户名！**");**  } if (pwd == null || pwd.equals("")) { **this.addFieldError("loginUser.password", "**请输入您的密码！**");**  }  }  } |

1. 修改 struts.xml 文件，在 Action 的配置中增加 validate()方法校验出错时的页面导航（<result name="input">），代码片段如下：

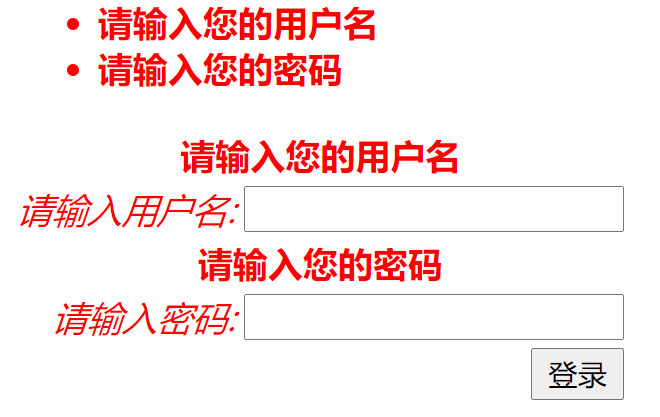
|  |
| --- |
| <struts>  <package name="strutsBean" extends="struts-default" namespace="/">  <action name="login" class="cn.edu.zjut.action.UserAction" method="login">  <result name="success">/loginSuccess.jsp</result>  <result name="fail">/loginFail.jsp</result>  **<result name="input">/login.jsp</result>** </action>  ……  </package>  </struts> |

1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；

用户名密码均不填



1. 修改 login.jsp 页面，在表单前增加 fielderror 标签：<s:fielderror/>，再通过浏览器访问 login.jsp 页面，观察并记录运行结果；



1. 修改 UserAction.java，在调用登录逻辑的 login()方法中，对登录情况进行校验：若登录成功，使用 addActionMessage()方法添加“登录成功！”的 Action 提示消息，若登录失败，使用 addActionError()方法添加 Action 级别的错误信息

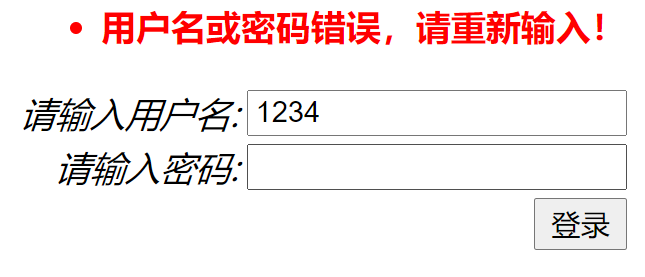
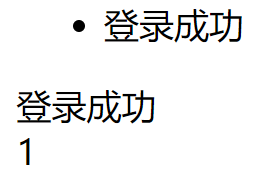
（代码片段如下）；

|  |
| --- |
| public class UserAction extends ActionSupport {  ……  public String login() {  UserService userServ = new UserService(); if (userServ.login(loginUser)) { **this.addActionMessage("**登录成功！**");** return "success";  } else { **this.addActionError("**用户名或密码错误，请重新输入！**")**; return "fail";  }  } |

1. 修改 login.jsp 页面，增加 actionerror 标签（<s:actionerror/>）Action 级别的错误信息；修改 loginSuccess.jsp，使用 actionmessage 标签（<s:actionmessage/>）

显示 Action 提示消息；

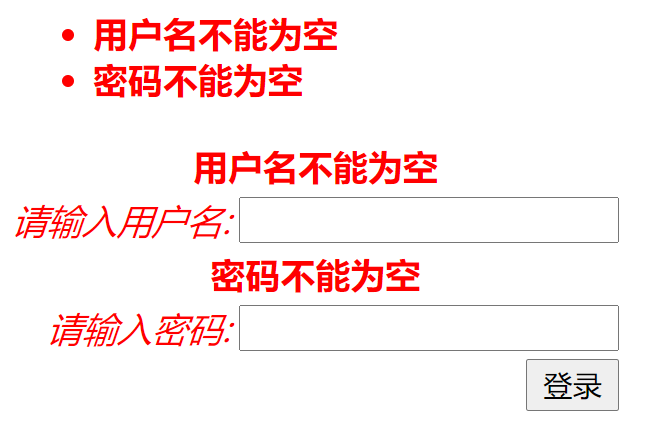
1. 修改 struts.xml 文件中用户登录的页面导航设置，将登录失败时转向的页面从 loginFail.jsp 修改为 login.jsp；
2. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；



1. 在工程 struts-prj2 中创建“UserAction-login-validation.xml”校验规则文件，使其与 UserAction 类位于同一目录下，配置校验信息，使用校验器对请求参数进行校验（代码如下）；

|  |
| --- |
| <?xml version="1.0" encoding="UTF-8"?>  <!DOCTYPE validators PUBLIC  "-//Apache Struts//XWork Validator 1.0.2//EN"  "http://struts.apache.org/dtds/xwork-validator-1.0.2.dtd">  <validators>  <field name="loginUser.account">  <field-validator type="requiredstring">  <param name="trim">true</param>  <message>用户名不能为空</message>  </field-validator>  </field>  <field name="loginUser.password">  <field-validator type="requiredstring">  <param name="trim">true</param>  <message>密码不能为空</message>  </field-validator>  </field>  </validators> |

1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；



1. 参考“实验二 Struts 基础应用”中的扩展实验，将 login.jsp、loginSuccess.jsp、 loginFail.jsp 三个页面进行国际化处理，把需要进行国际化的内容以键值对的形式写入资源文件 message\_zh\_CN.properties 和 message\_en\_US.properties

中；

1. 在资源文件中添加校验信息的键值对，并使用 native2ASCII 工具，将 message\_zh\_CN.properties 重新编码，将中文字符都转化为 unicode 码（代码

片段如下）；

|  |
| --- |
| #message\_en\_US.properties #field error message login.account.null=Please input your account! login.password.null=Please input your password!  #action error message login.error= Account or password error, please input again!  #action message login.success=Login successfully! |

1. 在工程 struts-prj2 的 src 目录中创建 struts.properties 文件，通过它加载资源文件（具体代码如下）；

**struts.custom.i18n.resources=cn.edu.zjut.local.message** struts.i18n.encoding=GBK

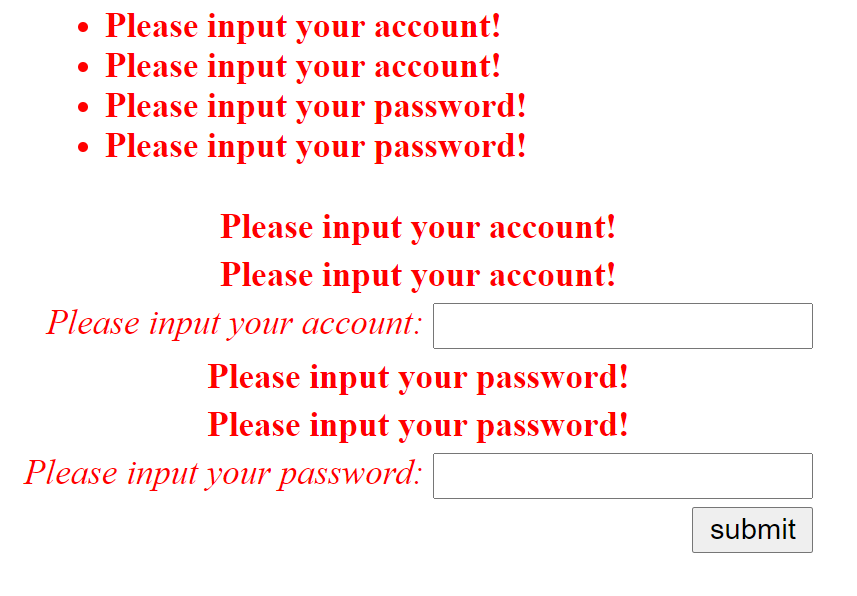
1. 修改 UserAction.java，使用 ActionSupport 类的 getText()方法，获取国际化资源文件中的信息（代码片段如下）；

|  |
| --- |
| public class UserAction extends ActionSupport {  ……  public String login() {  UserService userServ = new UserService(); if (userServ.login(loginUser)) { this.addActionMessage(**this.getText("login.success")**); return "success";  } else {  this.addActionError(**this.getText("login.error")**); return "fail";  } public void validate() {  String account = loginUser.getAccount(); String pwd = loginUser.getPassword(); if (account == null || account.equals("")) { this.addFieldError("loginUser.account", **this.getText("login.account.null")**);  } if (pwd == null || pwd.equals("")) { this.addFieldError("loginUser.password", **this.getText("login.password.null")**);  }  }  } |

1. 修改 UserAction-login-validation.xml，获取国际化资源文件中的信息（代码片段如下）；

|  |
| --- |
| <validators>  <field name="loginUser.account">  <field-validator type="requiredstring">  <param name="trim">true</param>  <message key="login.account.null"/>  </field-validator>  </field>  ……  </validators> |

1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；



1. 修改 UserBean.java，将用于保存注册用户生日的变量类型改为 Date 类型，使用 Struts2 内置的类型转换器对请求参数进行校验；
2. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页面，当用户输入的生日不合法时，观察并记录运行结果；



1. 在工程 struts-prj2 的 cn.edu.zjut.action 包中创建局部属性文件 “UserAction.properties”，修改类型转换的校验信息，并使用 native2ASCII 工具将 UserAction.properties 重新编码（代码片段如下）；

#其中invalid.fieldvalue不能随意修改，loginUser.birthday是请求参数域名，

#应根据实际需要进行修改

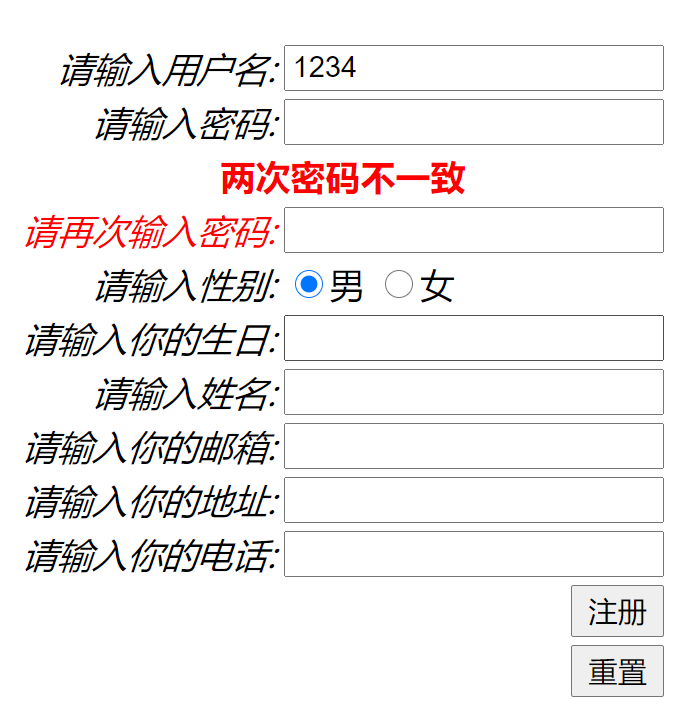
invalid.fieldvalue.loginUser.birthday=生日必须是日期，并符合

“yyyy-mm-dd”格式

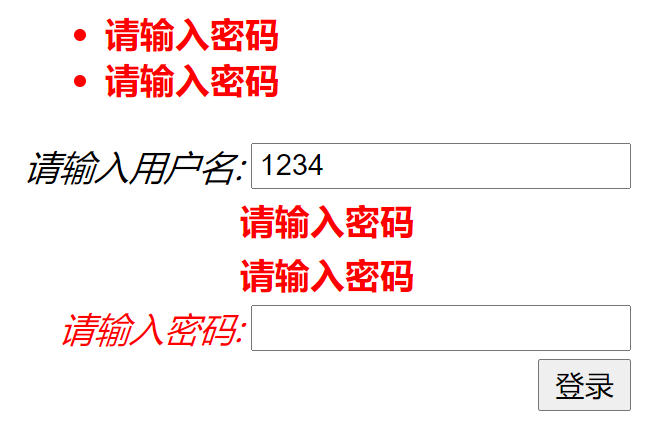
1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页面，当用户输入的生日不合法时，观察并记录运行结果；

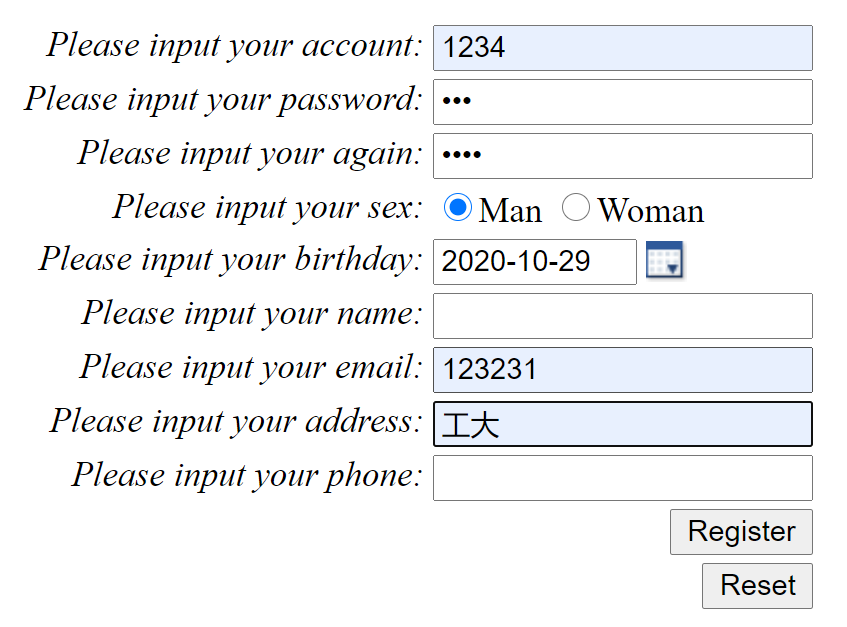


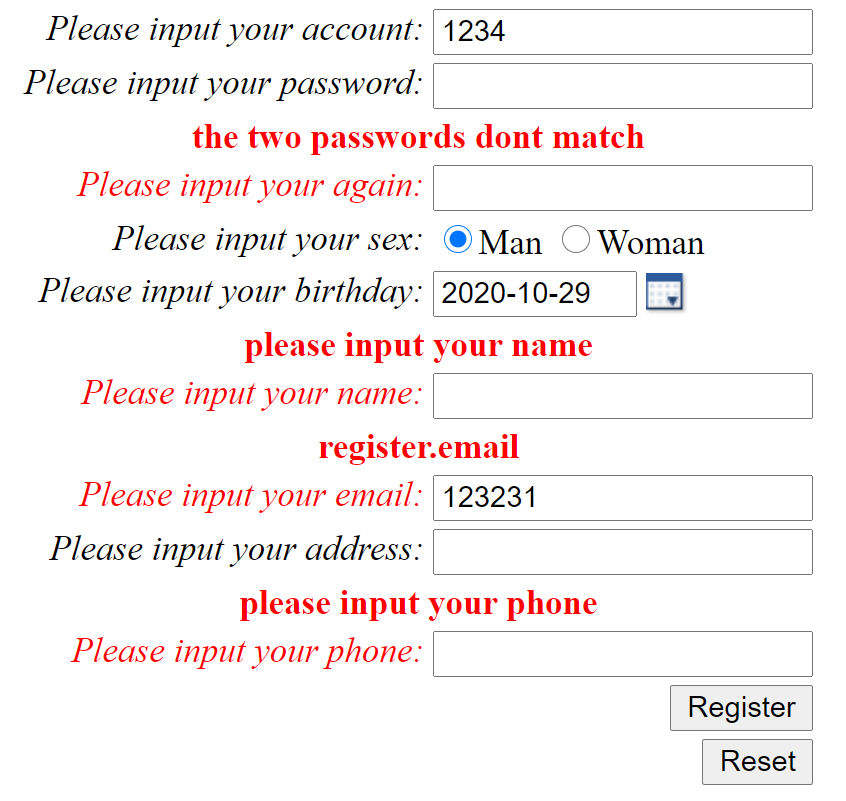
1. 参考实验步骤 9，在工程 struts-prj2 的 cn.edu.zjut.action 包中创建 UserAction -register-validation.xml 文件，增加校验信息的配置，使用校验器对用户注册的请求参数进行校验，要求注册时两次密码输入相同、email 地址格式符合要求等；
2. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页面，观察并记录运行结果；



1. 修改 UserAction 类，将 validate()的方法名改为 validateLogin()，并增加 validateRegister()方法，参考实验步骤 1，使用手工编码方式对请求参数进行数据校验；
2. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 register.jsp 页面，观察并记录运行结果。







# （四）实验要求

1、填写并上交实验报告，报告中应包括：

1. 运行结果截图；
2. 根据实验步骤 1-7 和步骤 21-22，查找相关资料，总结 Action 类中 validate() 方法和 validateXxx()方法的作用、使用时的要点或注意事项，总结在 JSP 页面中显示错误信息和提示信息的方法，并记录实验中相应的关键代码；

Validate校验器方法最先被默认调用，validateXxx()默认在Xxx()方法执行前校验

1. 根据实验步骤 8-9、19-20，总结使用校验器校验的方法；在 Struts2 的核心包 xwork-core-2.3.15.1.jar\com\opensymphony\xwork2\validator\validators 路径下找到 default.xml 文件，查找相关资料，总结校验规则文件中主要元素的作用和配置方法，并结合相应案例将其记录下来；

int 整数；  
double 实数；  
date 日期；  
expression  两数的关系比较；   
email Email地址；  
url  
visitor  
conversion   
regex 正则表达式验证；  
required 是否为空；  
requiredstring 必须字符；  
stringlength 输入字符长度限制;

1. 根据实验步骤 10-14，总结在 Action 中使用国际化资源文件的步骤及方法，并结合实验中相应的关键代码将其记录下来；

在国际化文件中添加key，value信息

Action中用this.getText(key)获取国际化信息

1. 根据实验步骤 15-18，总结 Struts2 中常用的内置类型转换器及其使用方法；（6）碰到的问题及解决方案或思考；（7）实验收获及总结。

收获：用gradle构建项目时，校验器的配置文件均放在resources下面，按照action的文件目录构建，但要注意的是应按包的结构一级级构建文件夹

2、上交程序源代码，代码中应有相关注释。三、扩展实验——**Action** 类与 **ServletAPI**

# （一）实验目的

1. 掌握在 Action 中访问 Servlet API 的四种方法，理解四种方法的区别；
2. 进一步理解 Action 与 Servlet 的区别；
3. 进一步熟悉 Struts2 标签的使用方法；

# （二）基本知识与原理

1. Struts2 框架中的 Action 类没有与任何 Servlet API 耦合，因此 Action 类可以脱离 Servlet 容器环境进行单元测试；
2. ActionContext 是 com.opensymphony.xwork2 包中的一个类，该类表示一个 Action 运行时的上下文；
3. 当 Action 类需要通过请求、会话或上下文存取属性时，可以通过 ActionContext 类完成，也可以通过实现 Struts 提供的接口：RequestAware、SessionAware 和 ApplicationAware 完成，而不必调用 Servlet API 中的 HttpServletRequest、 HttpSession 和 ServletContext 对象，从而保持 Action 与 Servlet API 的解耦；
4. 在 Action 类中直接访问 Servlet API，可以通过实现 Struts2 提供的接口：

ServletContextAware、ServletRequestAware、ServletResponseAware 完成，也可以通过 ServletActionContext 工具类实现，但 Action 将与 Servlet API 直接耦合。

# （三）实验内容及步骤

1. 在 struts-prj2 中修改 UserAction 类，通过 ActionContext 获取请求、会话和上下文对象相关联的 Map 对象来实现存取属性的功能（代码片段如下）；

|  |
| --- |
| public class UserAction extends ActionSupport {  ……  private Map request,session,application; public String login() {  //获取ActionContext对象 |
| **ActionContext ctx= ActionContext.getContext();**  //通过ActionContext对象获取请求、会话和上下文对象相关联的Map对象  **request=(Map) ctx.get("request"); session=(Map) ctx.getSession(); application=(Map) ctx.getApplication();**  //访问application范围的属性值  **Integer counter = (Integer)application.get("counter");** if(counter==null) counter=1;  else counter=counter+1;  //设置application范围的属性  **application.put("counter", counter);** UserService userServ = new UserService(); if (userServ.login(loginUser)) {  //设置session范围的属性  **session.put("user", loginUser.getAccount());**  //设置request范围的属性 **request.put("tip", "**您已登录成功**");** return "success";  } else {  return "fail";  }  }  } |

1. 修改 loginSuccess.jsp 页面，从请求、会话和上下文对象中获取属性值并显示在页面中（代码片段如下）；

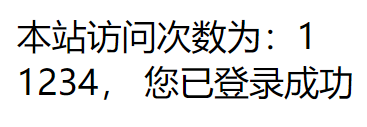
<body> 本站访问次数为：<s:property value="#application.counter"/><br>

<s:property value="#session.user"/>，

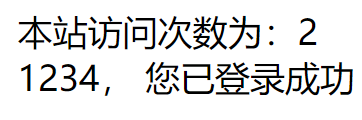
<s:property value="#request.tip"/>

</body>

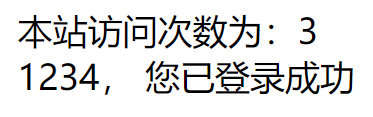
1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；



刷新



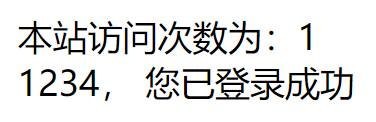
多次刷新



1. 修改 UserAction 类，通过实现 Struts 提供的接口：RequestAware、SessionAware 和 ApplicationAware，获取请求、会话和上下文对象相关联的 Map 对象来实现存取属性的功能（代码片段如下）；

|  |
| --- |
| public class UserAction extends ActionSupport **implements RequestAware,SessionAware,ApplicationAware**{  ……  private Map request,session,application; **public void setRequest(Map<String, Object> request) {** |
| **this.request=request;**  **} public void setSession(Map<String, Object> session) { this.session=session;**  **} public void setApplication(Map<String, Object> application) { this.application=application;**  **}**  public String login() {  Integer counter = (Integer)application.get("counter"); if(counter==null) counter=1;  else counter=counter+1;  application.put("counter", counter); UserService userServ = new UserService(); if (userServ.login(loginUser)) { session.put("user", loginUser.getAccount());  request.put("tip", "您已登录成功"); return "success";  } else {  return "fail";  }  }  } |

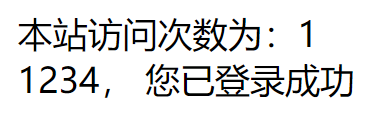
1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果；



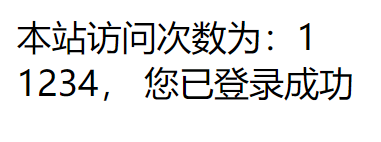
1. 修改 UserAction 类，查找相关资料，尝试通过接口：ServletContextAware、

ServletRequestAware、ServletResponseAware 直接访问 Servlet API，实现以上

实验步骤 1-3 的相同功能，重新运行并记录结果；



1. 修改 UserAction 类，查找相关资料，尝试通过 ServletActionContext 工具类直接访问 Servlet API，实现以上实验步骤 1-3 的相同功能，重新运行并记录结果；



1. 尝试利用 Servlet API 添加购物车功能，在工程 struts-prj2 的 cn.edu.zjut.bean 包中创建 Item.java 用于记录商品信息（代码片段如下）；

|  |
| --- |
| public class Item { private String itemID; private String name; private String description; private double cost;  public Item(String itemID, String name,  String description, double cost) { |
| setItemID(itemID); setTitle(name); setDescription(description); setCost(cost);  }  //省略getters/setters方法 } |

1. 在工程 struts-prj2 的 cn.edu.zjut.bean 包中创建 ItemOrder.java 用于记录购物车中的一条商品信息和购买数量（代码片段如下）；

|  |
| --- |
| public class ItemOrder { private Item item; private int numItems;  public ItemOrder(Item item) { setItem(item); setNumItems(1);  }  //省略getters/setters方法 } |

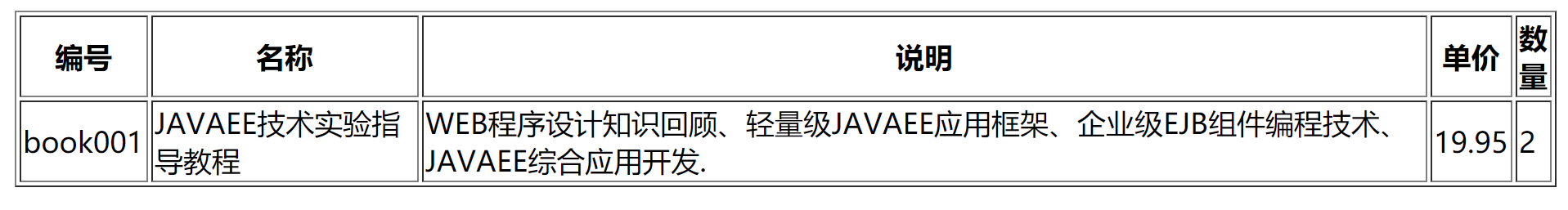
1. 在工程 struts-prj2 的 cn.edu.zjut.bean 包中创建 ShopppingCart.java 用于记录用户的购物车信息，为简化操作，在购物车的构造函数中加入商品信息（代码片段如下）；

|  |
| --- |
| public class ShoppingCart { private List itemsOrdered; public ShoppingCart() { itemsOrdered = new ArrayList();  Item item = new Item("book001", "JAVAEE技术实验指导教程",  "WEB程序设计知识回顾、" + "轻量级JAVAEE应用框架、"  +"企业级EJB组件编程技术、"+"JAVAEE综合应用开发.",19.95); ItemOrder itemorder = new ItemOrder(item); itemorder.setNumItems(2); itemsOrdered.add(itemorder);  } public List getItemsOrdered() { return (itemsOrdered);  } public synchronized void addItem(String itemID) {} public synchronized void setNumOrdered(String itemID, int numOrdered) {}  } |

1. 修改 UserAction 类的 login 方法，当登录成功时，创建 ShoppingCart 对象并存入会话中；
2. 修改 loginSuccess.jsp 页面，从会话中获取购物车信息并显示在页面中（代码片段如下）；

|  |
| --- |
| <body>  <table border=1>  <s:iterator value="#session.shoppingcart.itemsOrdered" >  <tr>  <th>编号</th><th>名称</th><th>说明</th><th>单价</th><th>数量</th>  </tr> <tr>  <td><s:property value="item.itemID"/></td>  <td><s:property value="item.name"/></td>  <td><s:property value="item.description"/></td>  <td><s:property value="item.cost"/></td>  <td><s:property value="numItems"/></td>  </tr>  </s:iterator>  </table>  </body> |

1. 重新将 struts-prj2 部署在 Tomcat 服务器上，通过浏览器访问 login.jsp 页面，观察并记录运行结果。



# （四）实验要求

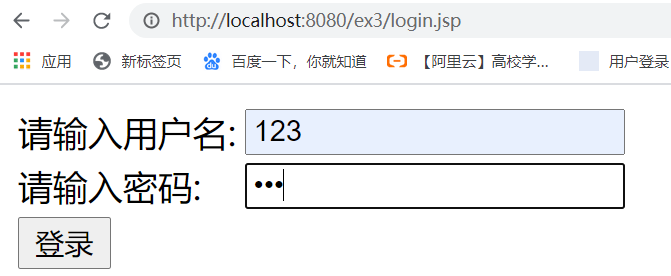
1、填写并上交实验报告，报告中应包括：

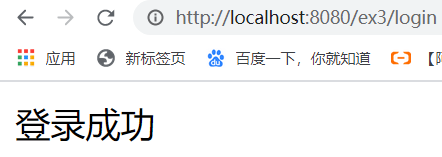
1. 运行结果截图；
2. 根据实验过程，观察 Action 中访问 Servlet API 的四种方法，总结四种方法的区别，并记录下来；
3. 整理购物车功能相关的关键代码，记录相应的运行结果或报错信息；
4. 碰到的问题及解决方案或思考；
5. 实验收获及总结。
6. 上交程序源代码，代码中应有相关注释。

实验一运行结果

**在struts.xml文件中通过method属性指定方法名**

登录





注册





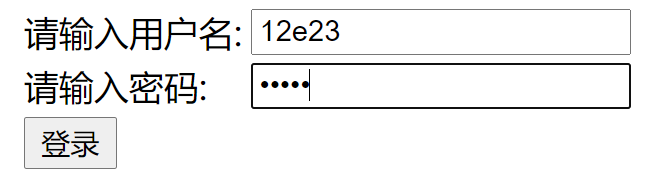
**使用动态方法调用方式（DMI）**

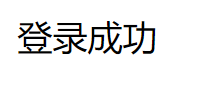
注册





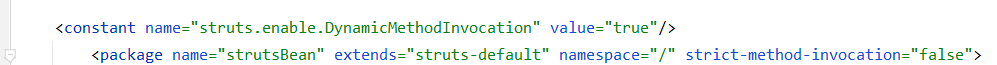
登录



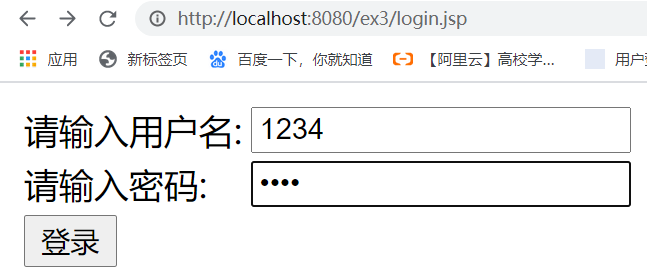


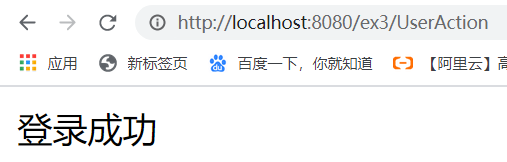
**使用提交按钮的method属性**

注意：由于struts2.5版本问题，对安全性做了优化，需要添加配置代码如图



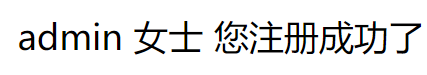
登录



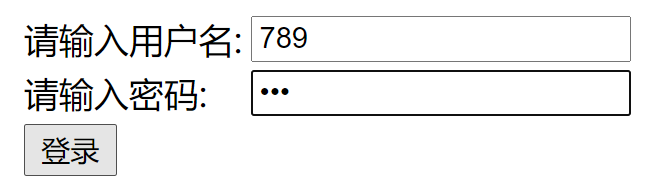


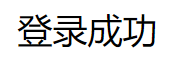
注册



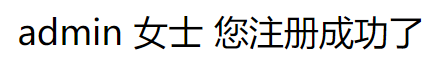


**使用通配符配置Action**

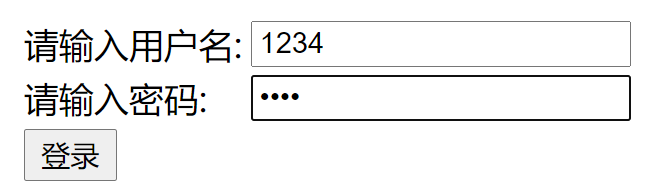


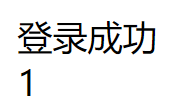




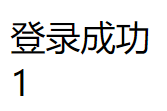


第一次登录

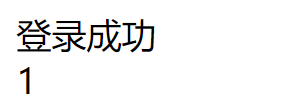




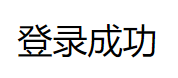
刷新



刷新



修改为type=”redirect”



实验二运行结果