Java EE 企业应用系统开发
MVC

王晓东
wangxiaodong@ouc.edu.cn

中国海洋大学

December 22, 2017

References

1. http://struts2.group.iteye.com/group/wiki/?category_id=44
2. 轻量级 Java EE 企业应用实战

大纲

为什么要学习框架？

- Struts2
- Spring
- Hibernate

为什么要学习框架？框架到底给我带来了什么？

接下来…

JSP 方式

JSP 在 HTML 代码里写 Java 代码完成业务逻辑。

```
<%
    String name = request.getParameter("name");
    String password = request.getParameter("password");

    UserHandler userHandler = new UserHandler();
    if(userHandler.authenticate(name, password)) {
%>
<p>Congratulations, login successfully. </p>
<%
     } else {
%>
<p>Sorry, login failed.</p>
<%
     }
%>
```

JSP 方式

❖ 仅有的一点优势

1. 无需额外的配置文件，无需框架的帮助，即可完成逻辑。
2. 简单易上手。

❖ 劣势

1. Java 代码由于混杂在一个 HTML 环境中而显得混乱不堪，可读性非常差。一个 JSP 文件有时候会变成几十 K，甚至上百 K，经常难以定位逻辑代码的所在。
2. 编写代码时非常困惑，不知道代码到底应该写在哪里，也不知道别人是不是已经曾经实现过类似的功能，到哪里去引用。
3. 突然之间，某个需求发生了变化。于是，每个人蒙头开始全程替换，还要小心翼翼的，生怕把别人的逻辑改了。
4. 逻辑处理程序需要自己来维护生命周期，对于类似数据库事务、日志等众多模块无法统一支持。

JSP 方式

❖ 仅有的一点优势

1. 无需额外的配置文件，无需框架的帮助，即可完成逻辑。
2. 简单易上手。

❖ 劣势

1. Java 代码由于混杂在一个 HTML 环境中而显得混乱不堪，可读性非常差。一个 JSP 文件有时候会变成几十 K，甚至上百 K，经常难以定位逻辑代码的所在。
2. 编写代码时非常困惑，不知道代码到底应该写在哪里，也不知道别人是不是已经曾经实现过类似的功能，到哪里去引用。
3. 突然之间，某个需求发生了变化。于是，每个人蒙头开始全程替换，还要小心翼翼的，生怕把别人的逻辑改了。
4. 逻辑处理程序需要自己来维护生命周期，对于类似数据库事务、日志等众多模块无法统一支持。

JSP 方式

❖ 仅有的一点优势

1. 无需额外的配置文件，无需框架的帮助，即可完成逻辑。
2. 简单易上手。

❖ 劣势

1. Java 代码由于混杂在一个 HTML 环境中而显得混乱不堪，可读性非常差。一个 JSP 文件有时候会变成几十 K，甚至上百 K，经常难以定位逻辑代码的所在。
2. 编写代码时非常困惑，不知道代码到底应该写在哪里，也不知道别人是不是已经曾经实现过类似的功能，到哪里去引用。
3. 突然之间，某个需求发生了变化。于是，每个人蒙头开始全程替换，还要小心翼翼的，生怕把别人的逻辑改了。
4. 逻辑处理程序需要自己来维护生命周期，对于类似数据库事务、日志等众多模块无法统一支持。

JSP 方式

❖ 仅有的一点优势

1. 无需额外的配置文件，无需框架的帮助，即可完成逻辑。
2. 简单易上手。

❖ 劣势

1. Java 代码由于混杂在一个 HTML 环境中而显得混乱不堪，可读性非常差。一个 JSP 文件有时候会变成几十 K，甚至上百 K，经常难以定位逻辑代码的所在。
2. 编写代码时非常困惑，不知道代码到底应该写在哪里，也不知道别人是不是已经曾经实现过类似的功能，到哪里去引用。
3. 突然之间，某个需求发生了变化。于是，每个人蒙头开始全程替换，还要小心翼翼的，生怕把别人的逻辑改了。
4. 逻辑处理程序需要自己来维护生命周期，对于类似数据库事务、日志等众多模块无法统一支持。

需求的变化

在这个时候，如果有一种方式，它能够将页面上的那些 Java 代码抽取出来，让页面上尽量少出现 Java 代码，该有多好。

于是许多人开始使用 servlet 来处理那些业务逻辑。

Servlet 方式

```
 1  public class LoginServlet extends HttpServlet {
 2
 3    /* (non-Javadoc)
 4    * @see javax.servlet.http.HttpServlet#doPost(javax.servlet.http.HttpServletRequest,
 5    javax.servlet.http.HttpServletResponse)
 6    */
 7    @Override
 8    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
 9    throws ServletException, IOException {
10      String message = null;
11      RequestDispatcher dispatcher = req.getRequestDispatcher("/result.jsp");
12      String name = req.getParameter("name");
13      String password = req.getParameter("password");
14
15      UserHandler userHandler = new UserHandler();
16      if(userHandler.authenticate(name, password)) {
17        message = "恭喜你，登录成功";
18      } else {
19        message = "对不起，登录失败";
20      }
21
22      req.setAttribute("message", message);
23      dispatcher.forward(req, resp);
24    }
25  }
```

Servlet 方式

同时，我们需要在 web.xml 中为这个 servlet 配置 url 的请求映射关系。

```
1  <servlet>
2    <servlet-name>Login</servlet-name>
3    <servlet-class>com.demo2do.servlet.LoginServlet</servlet-class>
4  </servlet>

6  <servlet-mapping>
7    <servlet-name>Login</servlet-name>
8    <url-pattern>/Login</url-pattern>
9  </servlet-mapping>
```

框架方式

时代进一步发展，人们发现简单的 JSP 和 Servlet 已经很难满足人们懒惰的要求了。于是，人们开始试图总结一些公用的 Java 类，来解决 Web 开发过程中碰到的问题。这时，横空出世了一个框架，叫做 Struts。它非常先进地实现了MVC 模式，成为了广大程序员的福音。

在一定程度上，Struts 能够解决 Web 开发中的职责分配问题，使得显示与逻辑分开。不过开始的在很长一段时间里，学习使用 Struts 的程序员往往无法清晰的明白我们到底需要 Web 框架帮我们做什么，我们到底需要它完成点什么功能。

那么我们需要什么？

在回顾写代码的历史之后，回头来看看，我们到底需要什么？无论是使用 JSP，还是使用 Struts1，或是 Struts2，我们至少都需要一些必须的元素（如果没有这些元素，或许我还真不知道这个程序会写成什么样子）：

1. **数据**在这个例子中，就是 name 和 password。他们共同构成了程序数据的核心载体。事实上，我们往往会有一个 User 类来封装 name 和 password，这样会使得我们的程序更加 OO。无论怎么说，数据会穿插在这个程序的各处，成为程序运行的核心。
2. 页面展示
3. 处理具体业务的场所

那么我们需要什么？

在回顾写代码的历史之后，回头来看看，我们到底需要什么？无论是使用 JSP，还是使用 Struts1，或是 Struts2，我们至少都需要一些必须的元素（如果没有这些元素，或许我还真不知道这个程序会写成什么样子）：

1. 数据
2. **页面展示**在这个例子中，就是 login.jsp。没有这个页面，一切的请求、验证和错误展示也无从谈起。在页面上，我们需要利用 HTML，把我们需要展现的数据都呈现出来。同时我们也需要完成一定的页面逻辑，例如，错误展示，分支判断等。
3. 处理具体业务的场所

那么我们需要什么？

在回顾写代码的历史之后，回头来看看，我们到底需要什么？无论是使用 JSP，还是使用 Struts1，或是 Struts2，我们至少都需要一些必须的元素（如果没有这些元素，或许我还真不知道这个程序会写成什么样子）：

1. 数据
2. 页面展示
3. **处理具体业务的场所**不同阶段，处理具体业务的场所就不太一样。原来用 JSP 和 Servlet，后来用 Struts1 或者 Struts2 的 Action。

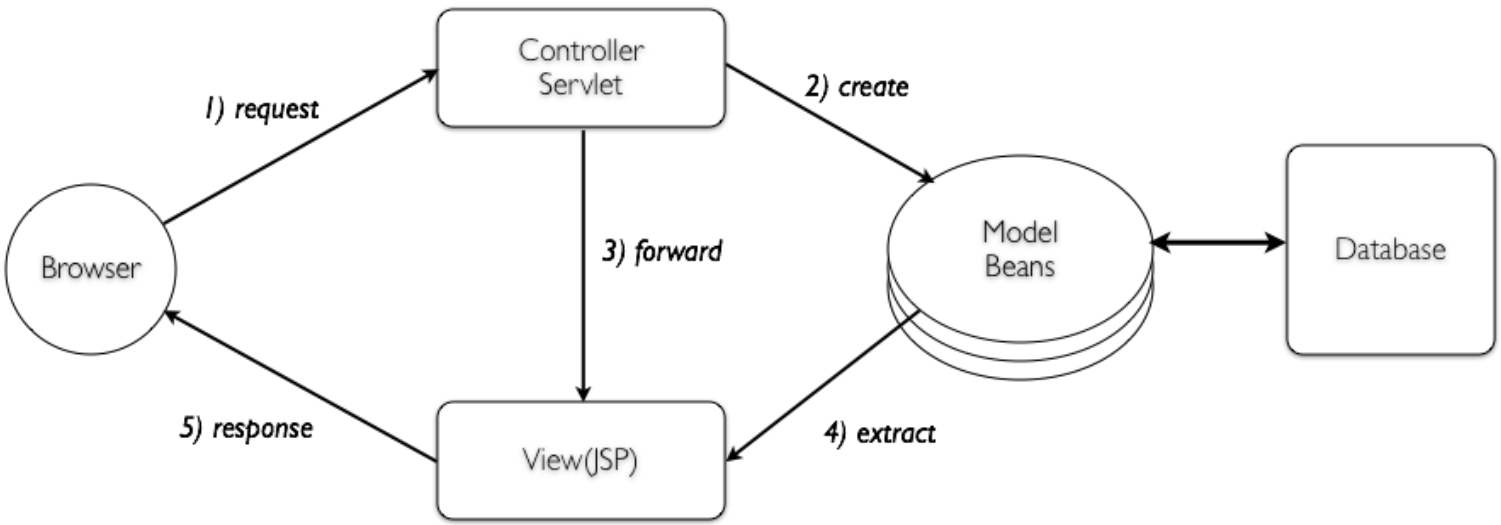
MVC

上面的这些必须出现的元素，在不同的时代被赋予了不同的表现形式，有的受到时代的束缚，其表现形式非常落后，有的已经不再使用。但是拨开这些外在的表现形式，我们就可以发现，这就是我们已经熟悉的 MVC。

- 数据 ➪ Model
- 页面展示 ➪ View
- 处理具体业务的场所 ➪ Control

框架不重要。只要能够深刻理解 MVC 的概念，框架只是几个 jar 包而已。

MVC



MVC Design Model

MVC 的特点

1. 多个视图可以对应一个模型，可以减少代码的复制，在模型发生改变时，易于维护。
2. 模型返回的数据与显示逻辑分离。模型数据可以应用任何显示技术，例如，使用 JSP、Velocity 模板或者直接产生 Excel。
3. 应用被分为三层，降低各层耦合，提高了可扩展性。
4. 控制层把不同模型和视图组合在一起，完成不同的请求，控制层包含了用户请求权限的概念。
5. MVC 符合软件工程化管理的思想，不同层各司其职，有利于通过工程化和工具化产生管理程序代码。

MVC

数据是动的，数据在 View 和 Control 层一旦运动起来，就会产生许多的问题：

- 数据从 View 层传递到 Control 层，如何使得一个个扁平的字符串，转化成一个个生龙活虎的 Java 对象。
- 数据从 View 层传递到 Control 层，如何方便的进行数据格式和内容的校验？
- 数据从 Control 层传递到 View 层，一个个生龙活虎的 Java 对象，又如何在页面上以各种各样的形式展现出来。
- 如果你试图将数据请求从 View 层发送到 Control 层，你如何才能知道你要调用的究竟是哪个类，哪个方法？一个 Http 的请求，又如何与 Control 层的 Java 代码建立起关系来？

MVC

数据是动的，数据在 View 和 Control 层一旦运动起来，就会产生许多的问题：

- 数据从 View 层传递到 Control 层，如何使得一个个扁平的字符串，转化成一个个生龙活虎的 Java 对象。
- 数据从 View 层传递到 Control 层，如何方便的进行数据格式和内容的校验？
- 数据从 Control 层传递到 View 层，一个个生龙活虎的 Java 对象，又如何在页面上以各种各样的形式展现出来。
- 如果你试图将数据请求从 View 层发送到 Control 层，你如何才能知道你要调用的究竟是哪个类，哪个方法？一个 Http 的请求，又如何与 Control 层的 Java 代码建立起关系来？

MVC

数据是动的，数据在 View 和 Control 层一旦运动起来，就会产生许多的问题：

- 数据从 View 层传递到 Control 层，如何使得一个个扁平的字符串，转化成一个个生龙活虎的 Java 对象。
- 数据从 View 层传递到 Control 层，如何方便的进行数据格式和内容的校验？
- 数据从 Control 层传递到 View 层，一个个生龙活虎的 Java 对象，又如何在页面上以各种各样的形式展现出来。
- 如果你试图将数据请求从 View 层发送到 Control 层，你如何才能知道你要调用的究竟是哪个类，哪个方法？一个 Http 的请求，又如何与 Control 层的 Java 代码建立起关系来？

MVC

数据是动的，数据在 View 和 Control 层一旦运动起来，就会产生许多的问题：

- 数据从 View 层传递到 Control 层，如何使得一个个扁平的字符串，转化成一个个生龙活虎的 Java 对象。
- 数据从 View 层传递到 Control 层，如何方便的进行数据格式和内容的校验？
- 数据从 Control 层传递到 View 层，一个个生龙活虎的 Java 对象，又如何在页面上以各种各样的形式展现出来。
- 如果你试图将数据请求从 View 层发送到 Control 层，你如何才能知道你要调用的究竟是哪个类，哪个方法？一个 Http 的请求，又如何与 Control 层的 Java 代码建立起关系来？

框架

框架是为了解决一个又一个在 Web 开发中所遇到的问题而诞生的。不同的框架，都是为了解决不同的问题，但是对于程序员而言，他们仅仅是 jar 包而已。框架的优缺点的评论，也完全取决于其对问题解决程度和解决方式的优雅性的评论。所以，**千万不要为了学习框架而学习框架，而是要为了解决问题而学习框架，这才是一个程序员的正确学习之道。**

接下来…

为 Web 应用增加 Struts 2 支持

1. 下载安装 Struts 2，登录 http://struts.apache.org/download.cgi，下载最新 Struts 2 的完整版（Full Distribution）。当前版本为：struts-2.3.15.1-all.zip。

```
[18:10]xiaodong@Wang:~/Installed/struts-2.3.15.1[0]
> ls
ANTLR-LICENSE.txt    OGNL-LICENSE.txt     apps
CLASSWORLDS-LICENSE.txt OVAL-LICENSE.txt docs
FREEMARKER-LICENSE.txt SITEMESH-LICENSE.txt lib
LICENSE.txt          XPP3-LICENSE.txt     src
NOTICE.txt           XSTREAM-LICENSE.txt
```

为 Web 应用增加 Struts 2 支持

2. 将 Struts 2 的 lib 目录下的**commons-fileupload-*.jar、commons-io-*.jar、freemarker-*.jar、javassist-*.jar、ognl-*.jar、struts2-core-*.jar、xwork-core-*.jar**必须类库复制到**Web 应用的 WEB-INF/lib 路径下**。如果需要在 DOS 或 Shell 窗口下手动编译 Strut 2 相关程序，还需要将 struts2-core-*.jar 和 xwork-core-*.jar 添加到系统的 CLASSPATH 环境变量。

为 Web 应用增加 Struts 2 支持

3. 编辑 Web 应用的 web.xml 配置文件，配置 Strut 2 的核心过滤器（Filter）。

```
1  <?xml version="1.0" encoding="GBK"?>
2  <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
3  xmlns="http://java.sun.com/xml/ns/javaee"
4  xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
5  xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
6  http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
7  id="WebApp_ID" version="3.0">

9  <!-- 定义 Struts2 的核心 Filter -->
10 <filter>
11 <filter-name>struts2</filter-name>
12 <filter-class>org.apache.struts2.dispatcher.ng.filter.
13               StrutsPrepareAndExecuteFilter</filter-class>
14 </filter>
15 <!-- 让Struts2 的核心 Filter 拦截所有请求 -->
16 <filter-mapping>
17 <filter-name>struts2</filter-name>
18 <url-pattern>/*</url-pattern>
19 </filter-mapping>
20 </web-app>
```

为 Web 应用增加 Struts 2 支持

4. 使用 Struts 2 的功能需要一个 struts.xml 配置文件，默认放在 Web 应用的类加载路径下（通常是 WEB-INF/classes）。

经过上述步骤，我们可以在一个 Web 应用中使用 Struts 2 的基本功能。

接下来…

创建并配置项目

在 Eclipse 中创建动态 Web 项目：Struts2Demo，按照上述步骤配置该项目。主要包括添加依赖的 Struts 2 类库，在 web.xml 中加入并配置核心过滤器。

增加登录处理

下面将为 Strut2Demo 应用增加一个简单的登录处理流程，以简要介绍 Struts 2 的开发步骤。

❖ 编写 JSP 页面

File: Struts2Demo/WebContent/login.jsp

```
1  <%@page language="java" contentType="text/html;␣charset=GBK"
2  pageEncoding="GBK"%>
3  <%@taglib prefix="s" uri="/struts-tags"%>
4  <!DOCTYPE html PUBLIC "-//W3C//DTD␣HTML␣4.01␣Transitional//EN"
5  "http://www.w3.org/TR/html4/loose.dtd">
6  <html>
7    <head>
8      <meta http-equiv="Content-Type" content="text/html;␣charset=GBK">
9      <title><s:text name="loginPage"/></title>
10   </head>
11   <body>
12     <s:form action="login">
13     <s:textfield name="username" key="user"/>
14     <s:textfield name="password" key="pass"/>
15     <s:submit key="login"/>
16     </s:form>
17   </body>
18 </html>
```

增加登录处理

上述 login.jsp 页面使用 Struts 2 标签库（后续介绍）定义了一个表单和三个简单表单域。几乎所有的 MVC 框架都会使用标签库，用以帮助开发者更加简单、更加规范的编写视图组件（例如 JSP 页面）。
提供 welcome.jsp 页面和 error.jsp 页面，作为登录成功、登录失败后的提示页面。
File: Struts2Demo/WebContent/welcome.jsp

```
...
<body>
  <s:text name="succTip">
    <s:param>${sessionScope.user}</s:param>
  /s:text><br/>
</body>
...
```

增加登录处理

File: Struts2Demo/WebContent/error.jsp

```
...
<body>
  <s:text name="failTip"/>
</body>
...
```

增加登录处理

为了让 Struts 2 运行起来，还必须为 Struts 2 框架提供一个配置文件：struts.xml。

File: Struts2Demo/src/struts.xml

```
1  <?xml version="1.0" encoding="GBK"?>
2  <!DOCTYPE struts PUBLIC
3  "-//Apache␣Software␣Foundation//DTD␣Struts␣Configuration␣2.1.7//EN"
4  "http://struts.apache.org/dtds/struts-2.1.7.dtd">
5  <!-- 指定 Struts 2 配置文件的根元素 -->
6  <struts>
7    <!-- 指定全局国际化资源文件 -->
8    <constant name="struts.custom.i18n.resources" value="mess"/>
9    <!-- 指定国际化编码所使用的字符集 -->
10   <constant name="struts.i18n.encoding" value="GBK"/>
11 </struts>
```

注意：在 Eclipse 的管理下，当 Eclipse 生成、部署 Web 项目时，会自动将 src 路径下除了 *.java 外所有的文件都复制到 Web 应用的 WEB-INF/classes 路径下，所以可以如上述创建 struts.xml 文件。

增加登录处理

上述 struts.xml 文件中制定了国际化资源文件的 base 名为 mess，所以需要为该应用提供一个 messa_zh_CN.properties 文件。

File: Struts2Demo/src/mess.properties

```
1 loginPage = 登录页面
2 errorPage = 错误页面
3 succPage = 成功页面
4 failTip = 对不起，您不能登录!
5 succTip = 欢迎， {0} ，您已经登录!
6 user = 用户名
7 pass = 密码
8 login = 登录
```

必须用 native2ascii 命令处理该国际化资源文件。

增加登录处理

前面定义了 login.jsp 页面中登录表单时指定改表单的 action 为 login，所以必须定义一个 Struts 2 的 Action，通常该继承 ActionSupport 基类。

增加登录处理

File: Struts2Demo/src/ouc/j2ee/action/LoginAction.java

```
1  ... ...
2  public class LoginAction extends ActionSupport {
3    // 定义封装请求参数的 username 和 password 属性
4    private String username;
5    private String password;
6    public String getUsername() {
7      return username;
8    }
9    public void setUsername(String username) {
10     this.username = username;
11   }
12   public String getPassword() {
13     return password;
14   }
15   public void setPassword(String password) {
16     this.password = password;
17   }
18   // 定义处理用户请求的 execute 方法
19   public String execute() throws Exception {
20     // 当username 为 oucj2ee，password 为 oucj2ee 时即登录成功
21     if (getUsername().equals("oucj2ee") && getPassword().equals("oucj2ee")) {
22       ActionContext.getContext().getSession().put("user", getUsername());
23       return SUCCESS;
24     } else {
25       return ERROR;
26     }
27   }
28 }
```

增加登录处理

增加 struts.xml 配置文件。

```
1  ... ...
2  <struts>
3    <!-- 指定全局国际化资源文件 -->
4    <constant name="struts.custom.i18n.resources" value="mess"/>
5    <!-- 指定国际化编码所使用的字符集 -->
6    <constant name="struts.i18n.encoding" value="GBK"/>
7    <!-- 所有的 Action 定义都应该放在 package 下 -->
8    <package name="oucj2ee" extends="struts-default">
9      <action name="login" class="ouc.j2ee.action.LoginAction">
10     <!-- 定义三个逻辑视图和物理资源之间的映射 -->
11       <result name="input">/login.jsp</result>
12       <result name="error">/error.jsp</result>
13       <result name="success">/welcome.jsp</result>
14     </action>
15   </package>
16 </struts>
```

配置一个名称为 login 的 Action，该 Action 配置三个 result 元素，用于指定逻辑视图与物理资源之间的映射。例如，当返回 input 逻辑视图名称时，系统跳转到/login.jsp 页面。

接下来…

❶ 在 web.xml 中配置核心过滤器

在 web.xml 文件中增加如下配置片段：

```
1  <!-- 定义 Struts 2 的核心过滤器 -->
2  <filter>
3    <filter-name>struts2</filter-name>
4    <filter-class>
5      org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter
6    </filter-class>
7  </filter>
8  <!-- 让 Struts 2 的核心过滤器拦截所有请求 -->
9  <filter-mapping>
10   <filter-name>struts2</filter-name>
11   <url-pattern>/*</url-pattern>
12 </filter-mapping>
```

❷ 定义包含表单数据的 JSP 页面

- 如果以 POST 方式提交表单数据，则定义包含表单数据的 JSP 页面。
- 如果仅仅以 GET 方式发送请求，则无需经过这一步。

❸ 定义处理用户请求的 Action 类

- Action 是 MVC 中的 C，即控制器。
- 控制器 Action 负责调用 Model 里的方法来处理请求。
- 在 Struts 2 中，MVC 框架控制器实际上由两个部分组成：拦截所有用户请求，处理请求的通用代码由核心控制器完成；实际业务控制则有 Action 处理。

注意：核心过滤器接收到用户请求后，通常会对用户请求进行简单预处理（例如解析、封装参数），然后通过反射来创建 Action 实例，并调用 Action 的指定方法（Struts 1 通常是 execute，Struts 2 可以是任意方法）来处理用户请求。

❹ 在 struts.xml 中配置 Action

通常采用如下 XML 片段配置 Action：

```
1 <action name="login" class="ouc.j2ee.action.LoginAction">
2   ... ...
3 </action>
```

上述配置指定如果用户请求的 URL 为 login，则使用 ouc.j2ee.action.
LoginAction 来处理。

注意：现在 Struts 2 的 Convension 插件借鉴 Rails 框架的优点，开始支持“约定优于配置”的思想，即采用约定的方式来规定用户请求地址和 Action 之间的对应关系。

❺ 配置处理结果和物理视图资源之间的对应关系

- 当 Action 处理用户请求结束后，通常会返回一个处理结果（通常使用简单的字符串），我们可以认为该名称是**逻辑视图名**。
- 逻辑视图名需要和指定的物理视图资源关联才有价值，所以我们需要配置处理结果之间的对应关系。

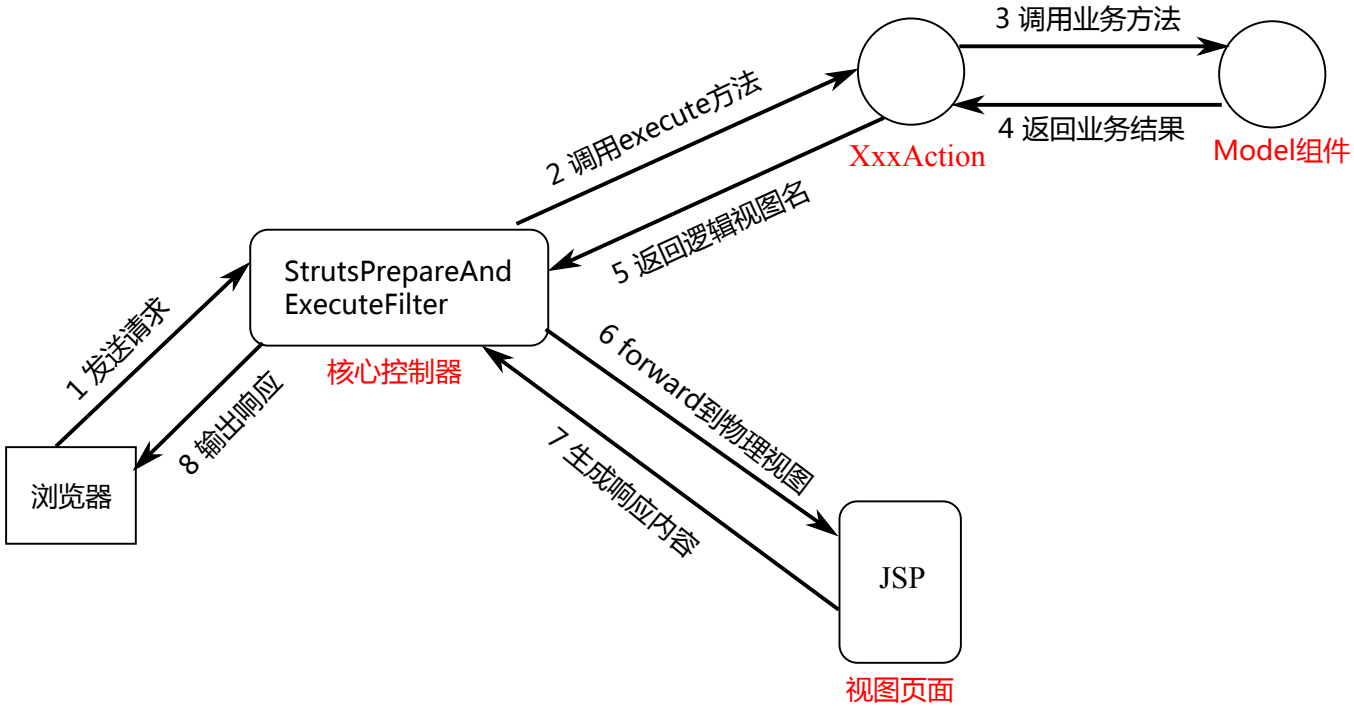
```
1 <action name="login" class="ouc.j2ee.action.LoginAction">
2   <!-- 定义3个逻辑视图和物理视图资源之间的映射 -->
3   <result name="input">/login.jsp</result>
4   <result name="error">/error.jsp</result>
5   <result name="success">/success.jsp</result>
6 </action>
```

❻ 编写视图资源

如果一个 Action 需要把一些数据传给视图资源，则可以借助OGNL 表达式。

经过上述步骤后，我们基本完成了一个 Struts 2 处理流程的开发，即可以完整的执行一次 HTTP 请求/响应过程。

Struts 2 流程



- StrutsPrepareAndExecuteFilter 和 XxxAction 共同构成 Strut 2 的控制器，其中前者称为核心控制器，后者称为业务控制器。
- 业务控制器并不与物理视图关联，这种做法提供了很好的解耦。
- 在 Struts 2 的控制下，用户请求不再向 JSP 页面发送，而是由核心控制器来“调用”JSP 页面来生成响应，此处调用不是直接调用，而是将请求 forward 到指定的 JSP 页面。

THE END

wangxiaodong@ouc.edu.cn