

# 轻量级 J2EE 框架应用

## E 2 A Simple Controller Based on Configuration File

学号: SA16225221

姓名: 欧勇

报告撰写时间: 2016/11/30

# 1. 主题概述

简要介绍主题的核心内容，如 MVC，及 MVC 中 Controller 的作用与实现  
作业内容：

1. 将 E1 中的控制器修改为基于配置文件的控制器。

1.1 将 E1 中编写的 Servlet 子类 LoginController，在 web.xml 中的配置修改为：可以拦截 “\*.saction” 类型的请求

1.2 在工程的 src 包下新建 controller.xml，在其中配置若干 action 与 result。示例如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<action-controller>
  <action>
    <name>login</name>
    <class>
      <name>water.servlet.action.LoginAction</name>
      <method>login</method>
    </class>
    <result>
      <name>success</name>
      <type>forward</type>
      <value>pages/success.jsp</value>
    </result>
    <result>
      <name>fail</name>
      <type>redirect</type>
      <value>pages/fail.jsp</value>
    </result>
  </action>
  <action>
    <name>register</name>
    <class>
      <name>water.servlet.action.RegisterAction</name>
      <method>login</method>
    </class>
    <!--other results -->
  </action>
  <!--other actions -->
</action-controller>
```

1.3 当一个 http request 请求访问 web container 资源时，由 LoginController 进行 request 拦截，解析请求，获取请求的 action。

1.4 LoginController 获取请求 action 后，解析 controller.xml（XML 解析，SAX、Dom 或其他），查找对应 name 的 action。如果在 controller.xml 中找到，则解析该 action 的配置，包括 class、result。如果没有找到，响应客户端信息为：不可识别的 action 请求。

1.5 LoginController 查找到 http request 请求的 action 资源后，利用其 class 属性实例化所指向的 action class (Java 反射机制, Reflection)，并执行指定的方法 action method。

1.6 action method 执行完毕后，返回字符串作为 result。

LoginController 根据该 action 配置的 result，查找匹配，将 result 指向的资源按 result type 所指定的方式返回到客户端。如果没有匹配的 result，响应客户端为信息为：没有请求的资源。

2. 比较 E1 与 E2 中的控制器，说明各自的优缺点及对 Struts 2 控制器的理解。

## 2. 假设

主题内容所参照的假设条件，或假定的某故事场景，如调试工具或软硬件环境

开发环境：

**Win10**

**Eclipse kepler**

**JDK 1.8**

**Tomcat 7.0**

### 3. 实现或证明

对主题内容进行实验实现，或例举证明，需描述实现过程及数据。如对 MVC 中 Controller 功能的实现及例证（图示、数据、代码等）

流程：

假设用户名为 world，密码为 hello

为了方便查看，采用 get 方式提交，可以通过浏览器 url 看到输入的用户名和密码（因为若采用 post 方式则无法通过 url 看到用户名和密码，所以采用 get 方式提交）

若登录成功则跳转 login\_success.jsp 页面，页面显示 Login Success 的字符串

若登录失败则跳转 login\_fail.jsp 页面，页面显示 Login Fail 的字符串

若使用未知的 action 提交，既 action="unknow.scaction" 则无法找到相应的方法处理，则跳转 error\_action.jsp 页面，页面显示“不可识别的 action 请求”提示字符串。

若返回的是未知的处理结果，则返回 error\_result.jsp 页面，页面显示“没有请求的资源”提示字符串

图 1：项目目录结构，可以看出项目名称为 SimpleController，src 文件夹下有一个 controller.xml 配置文件，其中记录有 action 和 result 相关的配置。

同时还有名为 me.king 的包，其下分别有两个类，一个是 LoginController 作为控制层，一个是 UserBean 作为模型层，然后还有 5 个 jsp 页面作为视图层，分别是 login\_fail.jsp，login\_success.jsp，error\_action.jsp，error\_result.jsp 和 login.jsp。

注意在 WEB-INF/lib 下需要导入 dom4j 的 jar 包，若仅仅只是将 jar 包放入 Java Resources/Libraries 中，则在编译时能通过，但是却无法完成处理，因为会在执行到语句 `new SAXReader()` 时报如下错误，提示找不到对应的类。

root cause

```
java.lang.NoClassDefFoundError: org/dom4j/io/SAXReader
    me.king.LoginController.doGet(LoginController.java:48)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:621)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:728)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:51)
```

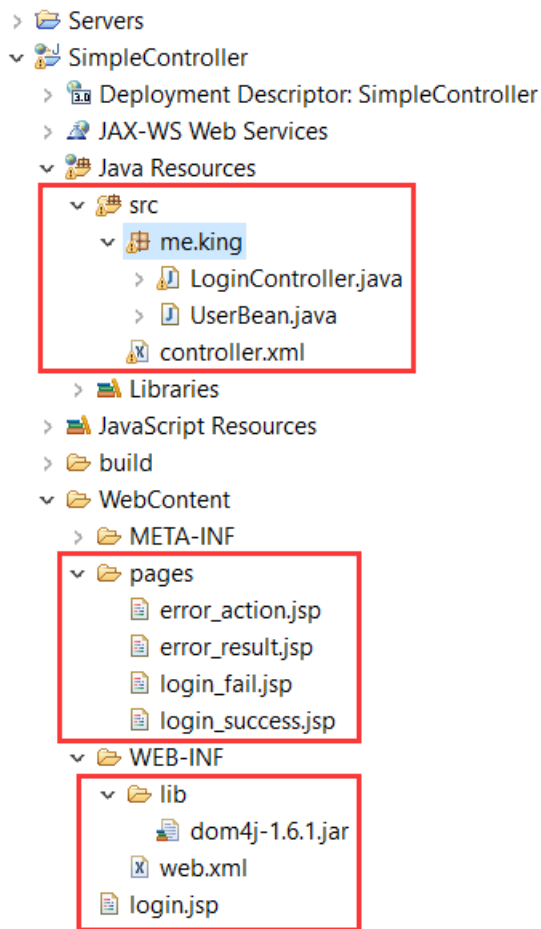


图 1: 项目目录结构

图 2.1: web.xml 配置文件截图

不使用注解的方式告知容器，而是用配置文件的方式配置控制器 LoginController 的映射路径,其中，将 login.jsp 配置为默认页面，将 servlet 控制层类 LoginController 映射名为同类名，同时，对所有以.scaction 结尾的 url 请求进行转发和控制。

```

LoginController.java login.jsp *web.xml controller.xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  <display-name>SimpleController</display-name>
  <welcome-file-list>
    <welcome-file>login.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>LoginController</servlet-name>
    <servlet-class>me.king.LoginController</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginController</servlet-name>
    <url-pattern>*.scaction</url-pattern>
  </servlet-mapping>
</web-app>
    
```

图 2.1: web.xml 配置文件截图

图 2.1: LoginController 代码截图

在 doGet 方法中进行控制转发，其他采用默认，若前端页面采用 post 方式提交，则在 doPost 方法中也需要进行转发处理，本次采用直接调用 doGet()方法进行处理。

```

1 package me.king;
2
3 import java.io.File;
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23 * Servlet implementation class LoginController
24 // @WebServlet("/LoginController")
25 public class LoginController extends HttpServlet {
26     private static final long serialVersionUID = 1L;
27
28
29
30 * @see HttpServlet#HttpServlet()
31 public LoginController() {
32     super();
33     // TODO Auto-generated constructor stub
34 }
35
36
37
38 * @see HttpServlet#doGet(HttpServletRequest request, HttpServletResponse response)
39 protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58 * @see HttpServlet#doPost(HttpServletRequest request, HttpServletResponse response)
59 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
60     // TODO Auto-generated method stub
61     doGet(request, response);
62 }
63
64 }

```

图 2.1: LoginController 代码截图

图 2.2: LoginController 类中 doGet 方法代码截图

doGet 中主要控制转发流程为：

1. 获取 request 中的 action 请求名，获取 controller.xml 配置文件
2. 解析配置文件并将所有的 action 标签都加入到 list 中
3. 设置标志位用于标志是否找到对应 action
4. 遍历 action 标签 list 并判断 name 标签和 method 标签
5. 利用反射机制实例化对应的类为 cls 对象
6. 调用实例化后 cls 的 method 方法并获取返回结果
7. 将返回结果与 result 标签 list 中的对应 name 对比
8. 获取对应 result 标签中的 value（返回地址），type（跳转还是内部重定向）
9. 执行返回操作

其中，try-catch 中的类型必须为 Exception，不能为 DocumentException，否则会在编译的时候报：Can't load IA 32-bit .dll on a AMD 64-bit platform 错误

java.lang.UnsatisfiedLinkError: D:\Program Files\apache-tomcat-7.0.47\bin\ntcnative-1.dll: Can't load IA 32-bit .dll on a AMD 64-bit platform

同时，在最后执行转发操作中，使用 response.sendRedirect()方法时，需要使用相对定位，否则会报：Path pages/login\_success.jsp does not start with a "/" character 错误，并且页面显示 404 错误码，找不到资源。

```

40= protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
41     // TODO Auto-generated method stub
42     ServletContext sc = getServletContext();
43     1 //获取请求的action
44     String actStr = request.getRequestURL().toString();
45     String actionName = actStr.substring(actStr.lastIndexOf('/')+1,actStr.indexOf(".saction")).toString();
46     InputStream is = this.getClass().getResourceAsStream("/me/king/controller.xml");//获取controller.xml配置文件,并以流的形式读入
47     try {
48         Document dc = (new SAXReader()).read(is);
49         2 List<Element> actions = dc.getRootElement().elements(); //获取所有的action标签并加入到list中
50         3 boolean isFindAction = false;
51         for (Element ele: actions){
52             4 if ( actionName.equals(ele.elementText("name"))){
53                 isFindAction = true; //将标志位设置为成功找到该action
54                 String className = ele.element("class").elementText("name"),
55                     classMethod = ele.element("class").elementText("method");
56
57                 //利用反射机制实例化对应的类
58                 5 Class<?> cls = Class.forName(className);
59                 //指定获取当前类的classMethod方法,同时指定参数列表的类型为string
60                 Method m = cls.getDeclaredMethod(classMethod,new Class[]{String.class,String.class});
61                 //调用该方法并将返回结果存入rstName中
62                 6 String rstName = (String) m.invoke(cls.newInstance(),request.getParameter("name"),request.getParameter("pwd"));
63
64                 List<Element> rst = ele.elements("result");//所有的result标签
65                 for (Element rstEle: rst){ //一个rstEle对应一个result节点
66                     7 if (rstName.equals( rstEle.elementText("name"))){ //当存在对应result返回结果的配置信息的时候
67                         String rstValue = rstEle.elementText("value");
68                         8 if (rstEle.elementText("type").equals("forward"))
69                             9 → sc.getRequestDispatcher(rstValue).forward(request, response); //forward为内部重定向
70                             → else response.sendRedirect(''+rstValue); //重定向根据相对地址跳转
71                         }else response.sendRedirect("./pages/error_result.jsp"); //不存在对应result的时候跳转至error_result页面
72                     }
73                 }
74             }
75             if (!isFindAction) //当找不到对应action请求的时候跳转error_action页面
76                 → response.sendRedirect("./pages/error_action.jsp");
77         } catch (Exception e) {
78             // TODO Auto-generated catch block
79             e.printStackTrace();
80         }

```

图 2.2: LoginController 类中 doGet 方法代码截图



```

LoginController.java login.jsp web.xml controller.xml
<?xml version="1.0" encoding="UTF-8"?>
<action-controller>
  <action>
    <name>login</name>
    <class>
      <name>me.king.UserBean</name>
      <method>checkLogin</method>
    </class>
    <result>
      <name>success</name>
      <type>forward</type>
      <value>/pages/login_success.jsp</value>
    </result>
    <result>
      <name>fail</name>
      <type>redirect</type>
      <value>/pages/login_fail.jsp</value>
    </result>
  </action>
  <action>
    <name>register</name>
  </action>
</action-controller>

```

图 2.3: controller.xml 配置文件截图

图 3: UserBean 代码截图,

定义一个简单的 `checkLogin` 方法, 参数为 `name` 和 `pwd`, 函数体内直接对比两个字符串, 若正确则返回“success”, 否则返回“fail”。同时为了测试方便, 当 `name` 为 `unknown` 并且 `pwd` 为 `result` 的时候返回“unknown”。

```

LoginController.java login.jsp web.xml controller.xml UserBean.java
1 package me.king;
2
3 public class UserBean {
4     public String checkLogin(String name, String pwd){
5         if(name.equals("unknown") && pwd.equals("result"))
6             return "unknown"; //此项判断仅用于测试返回unknown result
7
8         if(name.equals("world") && pwd.equals("hello"))
9             return "success";
10        else return "fail";
11    }
12 }

```

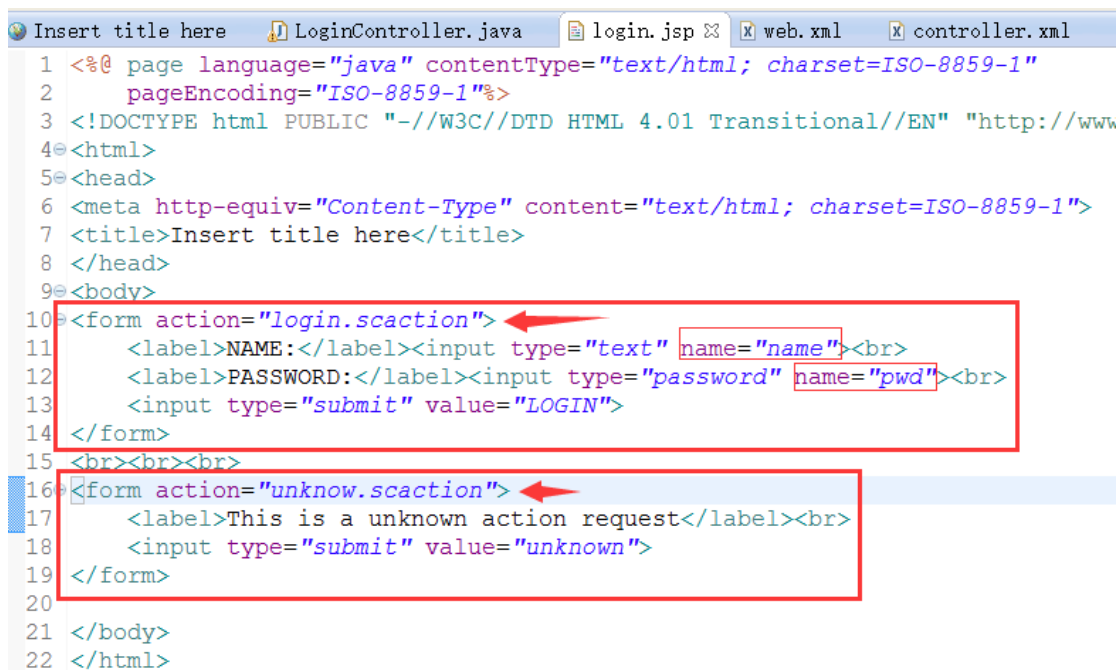
图 3: UserBean 代码截图

图 4.1: login.jsp 代码截图,

编写 2 个 form 表单, 第一个为正常登陆表单 action 设置为“login.scaction” (必须设置为后缀.scaction, 具体配置在 controller.xml 中已经设置完成), 设置用户名和密码的 `name`

属性分别为“name”和“pwd”。

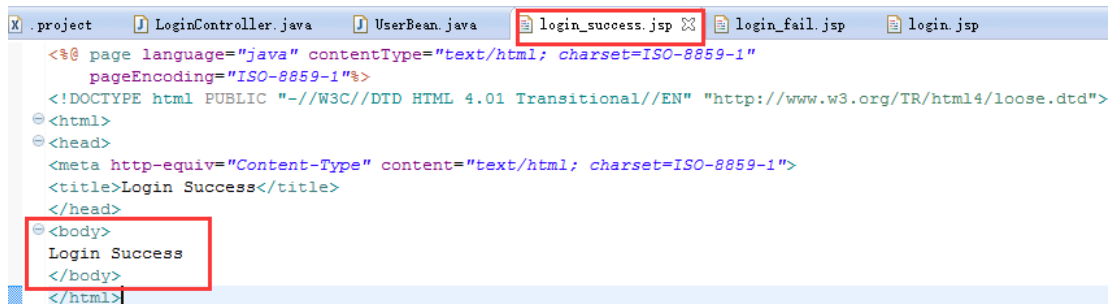
同时为了测试方便，定义一个 action 为“unknown.scaction”的表单，用于测试当 action 为未知的情况。



```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7 <title>Insert title here</title>
8 </head>
9 <body>
10 <form action="login.scaction">
11   <label>NAME:</label><input type="text" name="name"><br>
12   <label>PASSWORD:</label><input type="password" name="pwd"><br>
13   <input type="submit" value="LOGIN">
14 </form>
15 <br><br><br>
16 <form action="unknown.scaction">
17   <label>This is a unknown action request</label><br>
18   <input type="submit" value="unknown">
19 </form>
20
21 </body>
22 </html>
```

图 4.1: login.jsp 代码截图

图 4.2: login\_success.jsp 代码截图，  
简单的在 body 中写入 Login Success。



```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7 <title>Login Success</title>
8 </head>
9 <body>
10 Login Success
11 </body>
12 </html>
```

图 4.2: login\_success.jsp 代码截图

图 4.3: login\_fail.jsp 代码截图，  
简单的在 body 中写入 Login Fail。

```

project LoginController.java UserBean.java login_success.jsp login_fail.jsp login.jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login Fail</title>
</head>
<body>
Login Fail
</body>
</html>
    
```

图 4.3: login\_fail.jsp 代码截图

图 4.4: error\_action.jsp 代码截图

简单的在 body 中写入 Sorry, this is a unrecognized action request.

```

LoginController.java login.jsp web.xml controller.xml UserBean.java error_action.jsp
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>ERROR ACTION</title>
</head>
<body>
Sorry, this is a unrecognized action request.
</body>
</html>
    
```

图 4.4: error\_action.jsp 代码截图

图 4.5: error\_result.jsp 代码截图

简单的在 body 中写入 Sorry, there is no resources you request.

```

LoginController.java login.jsp web.xml controller.xml UserBean.java error_action.jsp error_result.jsp
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
4 <html>
5 <head>
6 <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
7 <title>ERROR RESULT</title>
8 </head>
9 <body>
10 Sorry, there is no resources you request.
11 </body>
12 </html>
    
```

图 4.5: error\_result.jsp 代码截图

图 5.1: 使用 Chrome 和 Eclipse 内置浏览器测试登录页面截图，可以看到 url 为: <http://localhost:8080/SimpleController/login.jsp>，可以在表单中分别填入 NAME 和 PASSWORD，点击 LOGIN 按钮提交至后台服务器。

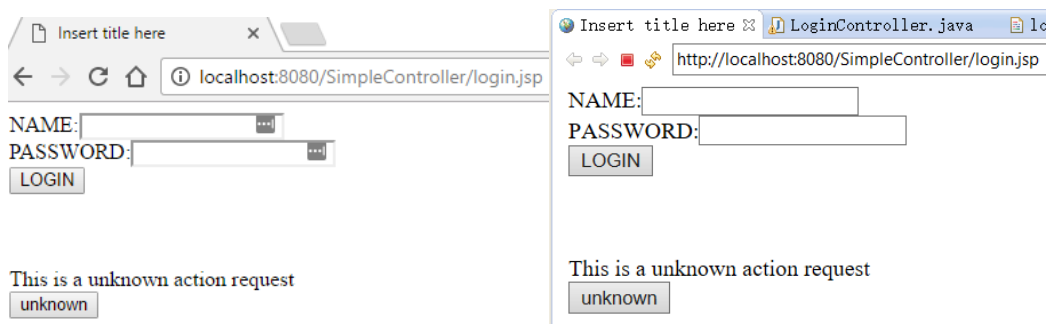


图 5.1: 使用 Chrome 和 Eclipse 内置浏览器测试登录页面截图

图 5.2: 登录成功截图,

可以通过 url 看到, 当输入的用户名 name 为 **world**, 密码 pwd 为 **hello** 时验证通过时显示 Login Success (虽然页面内容已经变为 login\_success.jsp 页面, 但此时 url 没有改变为 login\_success.jsp 而是在 login.saction 中)

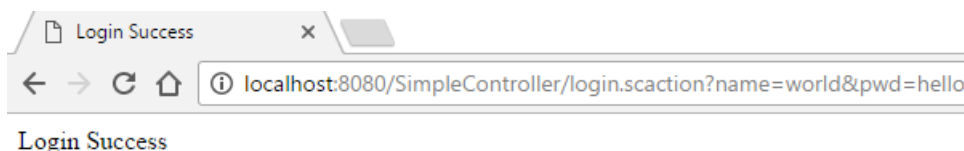


图 5.2: 登录成功截图

图 5.3: 登录失败截图,

可以通过 url 看到, 当输入的用户名 name 或密码 pwd 错误时, 验证失败, 跳转至 login\_fail.jsp 页面, 显示 Login Fail

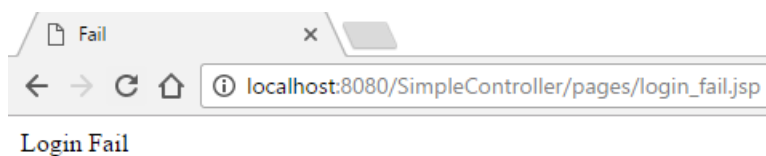


图 5.3: 登录失败截图

图 5.4: 不可识别的 action 请求截图

当点击图 5.1 中的 unknown 按钮的时候, 页面跳转至 error\_action.jsp 页面, 显示 Sorry, this is a unrecognized action request.

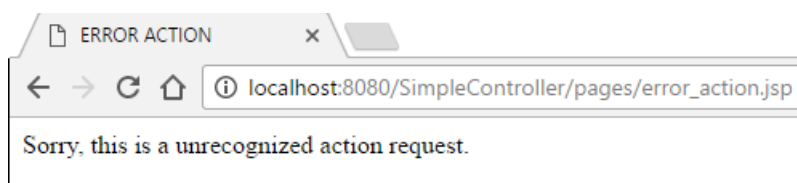


图 5.4: 不可识别的 action 请求截图

图 5.5: 没有请求的资源截图

当在登陆表单中输入的用户名 name 为 `unknown`，密码 pwd 为 `result` 时，跳转至 `error_result.jsp` 页面。显示 `Sorry, there is no resources you request.`

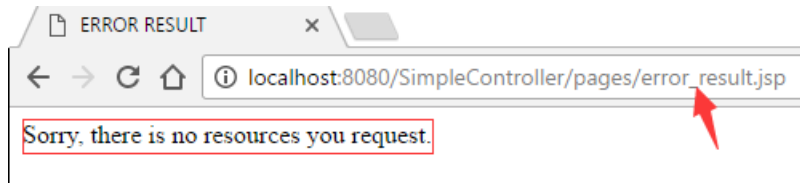


图 5.5: 没有请求的资源截图

## 4. 结论

对主题的总结，结果评论，发现的问题，或你的建议和看法。如 MVC 中 Controller 优点与缺点，个人看法（文字、图标、代码辅助等）

**比较 E1 与 E2 中的控制器，说明各自的优缺点及对 Struts 2 控制器的理解：**

最明显的区别就是：复杂度和灵活性上的差别。

**E1 的优点**在于复杂度(代码量和文件数目)小于 E2。

E1 采用注解的方式，省了 web.xml 配置文件的使用，同时将要转发的地址和处理请求的类采用硬编码的形式，绑定在处理逻辑的代码中了，比如类名处的 @WebServlet 注解和 UserBean 的实例化，以及最后的返回地址。

而 **E2 的优点**在于灵活修改，扩展方便，采用配置文件的形式则要灵活很多，虽然增加了代码量和复杂度，但是可以修改配置文件的内容以免每次都需要重新编译。

**关于 Struts2 控制器：**

网络上搜索关于 Struts2 控制器，有 FilterDispatcher 和 StrutsPrepareAndExecuteFilter，现在多使用后者。StrutsPrepareAndExecuteFilter 则分别可以分为 2 部分，一个是 prepare，表示准备，指 filter 中的 init 方法，配置的初始化，一个是 execute，指 doFilter 方法，执行过滤操作。

StrutsPrepareAndExecuteFilter 比 FilterDispatcher 好的地方在于：可以自定义 Filter 放在 prepare 和 execute 之间，先于 struts2 定义的过滤器执行，而这是在早期 FilterDispatcher 中无法做到的，因为放在 struts2 之后的自定义过滤器会失效。

控制器主要负责拦截所有用户请求，当用户的请求时以 .action 结尾的时候，则 Web Container 将该请求使用 struts2 框架处理。

与 Servlet 相比（我们自己定义并实现转发规则，控制处理流程），Struts2 的控制器可以不用显式的写 java 代码，而是在配置文件中配置 action 和 url 以及 location 等映射关系。虽然程序员使用 struts2 框架后不用显式编写 java 代码，但其实是因为这个工作被框架做了，它其实也是采用 servlet 来实现控制转发的。

ServletAction 就是这个 servlet 的名字，这个 servlet 的转发规则以及被映射到了 struts.xml 文件中。

**Struts2 的流程为：**

- 1.浏览器发送请求被 StrutsPrepareAndExecuteFilter 拦截
- 2.StrutsPrepareAndExecuteFilter 调用 xxxAction 的 execute 方法
- 3.xxxAction 调用 Model 组件的业务方法
- 4.Model 组件将处理结果返回给 xxxAction
- 5.xxxAction 将返回一个对应结果的逻辑视图名给 StrutsPrepareAndExecuteFilter
6. StrutsPrepareAndExecuteFilter 将转发 forward 到具体视图页面
- 7.视图页面生成响应内容返回给 StrutsPrepareAndExecuteFilter
- 8.最后 StrutsPrepareAndExecuteFilter 将输出响应结果给浏览器

## 5.参考文献

以上内容的理论知识点或技术点如果参考了网上或印刷制品，请在这里罗列出来

- [1] Java Reflection : <https://docs.oracle.com/javase/tutorial/reflect/>
- [2] XML Parser SAX : <http://www.saxproject.org/quickstart.html>
- [3] XML Parser DOM : [http://www.w3schools.com/dom/dom\\_parser.asp](http://www.w3schools.com/dom/dom_parser.asp)
- [4] 在 java 中使用 dom4j 解析 xml : <http://www.jb51.net/article/42323.htm>
- [5] 通过 Java 反射调用方法 : <http://blog.csdn.net/ichsonx/article/details/9108173>