轻量级 J2EE 框架应用

E 2　A Simple Controller Based on Configuration File

学号：SA17225052

姓名：戴赛

报告撰写时间：2017/12/10

1.主题概述

思路整理:

一上来就要登陆(主页兼登陆页面),然后手动点击注册进入注册页面(Html5 手动跳转)
注册页面进行注册(假装有赋值),注册成功跳回登陆(作业要求的是跳回欢迎界面,这里进行了修改)
用之前的进行登陆,登陆成功则显示欢迎界面(转发),登陆失败则返回失败界面(有按钮可以回登陆界面)

Controller
dopost 中:
获取 action 的名称
*解析 xml:SAX:
实现 contenHandler

获取 Action_name
获取 action_class // action 对应的类
获取 action_method // 对应类的方法

获取 result_name // 返回结果
获取 result_type // 返回类型(转发或重定向)
获取 result_value // 转发或重定向的地址页面

反射:
根据 req 得到 actionName 比对 action_class 找到 action 类,
根据 action_method 找到方法,并执行方法
根据方法结果比对 result_name 找到对应的 resu_value 执行 result_type

UseSC:
// 所有的类都在构造函数中进行输出提示
LoginAction/RegisterAction
只有一个方法 handle
Login 中根据传入的 name 与 pwd 进行判断,跳转,判断成功则返回 success,失败则返回 failure,判断条件直接写死
Register 中直接返回 success

controller.xml:

2.假设

上次的作业 UserSC 与 SimplController

3.实现或证明

1.讲控制器修改为基于配置的

1.1 修改 web.xml 使其能拦截*.sc

```
<servlet-mapping>
    <servlet-name>sc</servlet-name>
    <url-pattern>*.sc</url-pattern>
</servlet-mapping>
```

1.2 新建包并在其中新建相应 Action 类

LoginAction:

```
public class LoginAction {

    public String handleLogin(String name,String pwd) {
        //  Login 中根据传入的 name 与 pwd 进行判断,跳转,判断成功则返回 success
失败则返回 failure,内容写死

        if (name.equals("tom") && pwd.equals("123")) {
            return "success";
        } else {
            return "failure";
        }
    }
}
```

RegisterAction:

```
public class RegisterAction {

    public String handleRegister(String userName, String pwd) {
        // 直接返回 success
        return "success";
    }
}
```

1.3 Controller.xml 的配置

```
<controller>
    <action name="loginAction.sc" class="water.ustc.action.LoginAction"
method="handleLogin">
        <result name="success" type="forward" value="/welcome.jsp"></result>
        <result name="failure" type="redirect" value="/failure.jsp"></result>
    </action>
    <action name="registerAction.sc"
class="water.ustc.action.RegisterAction" method="handleRegister">
        <result name="success" type="forward" value="/login.jsp"></result>
```

```
    </action>
</controller>
```

1.4 doPost()中获取 Action 名称

使用 getServletPath()获得路径后再进行处理

```
String actionName = request.getServletPath();
String trueActionName = actionName.substring(1,actionName.length());
```

1.5 解析 controller.xml,使用 RAX 解析,重点是实现 contentHandler()接口

```
            SAXParserFactory saxParserFactory =
SAXParserFactory.newInstance();
            // 调用解析器
            SAXParser sp = saxParserFactory.newSAXParser();
            // 得到读取器
            XMLReader reader = sp.getXMLReader();
            // 实现 handler 接口
            ListHandler listHandler = new ListHandler(trueActionName);
            reader.setContentHandler(listHandler);
            // 绝对路径
            reader.parse("F:\\J2EE\\J2EETEst2\\UseSC\\src\\controller.xml");
            // resultList 里存储 result
            ArrayList resultList = listHandler.getResultName();
```

contentHandler()构造函数中读取到 actionName:

```
ListHandler(String name) {
    actionName = name;
    resultName = new ArrayList();
}
```

Handler 中的 startElement()方法,这个方法会在读取 xml 每一个含有标签的一行内容时调用一次,也没啥输出,但是会得到一个类似键值对的内容:

```
for(int i=0;atts!=null && i<atts.getLength();i++)
{
    String attName = atts.getQName(i); // 外部内容,如 name class type
    System.out.println("attName" + attName);
    String attValue = atts.getValue(i); // 引号内内容,如"loginAction.sc"
"forward"
    System.out.println("attValue" + attValue);

    if (attValue.equals(actionName)) {
        // 捕获到 action 的名称，进行 xml 解析。
        className = atts.getValue(i+1); // action_class
        method = atts.getValue(i+2); // action_method
        success = true; // xml 中有相应的 actionName,其实一定是相等的
    }

    // 内容是 result
    if (name.equals("result") && success) {
```

```
            result = new Result();
            result.setActionName(actionName);
            result.setResultName(atts.getValue(i));
            result.setResultType(atts.getValue(i + 1));
            result.setResultValue(atts.getValue(i + 2));
            resultName.add(result);
            break;
        }
    }
```

1.6　1.7 doPost()中实现反射,并用相应的方式调用方法

```
if (listHandler.isSuccess()) {
    // 反射
    Class actionClass = Class.forName(listHandler.getClassName());
    Object object = actionClass.newInstance();
    Method method =
actionClass.getDeclaredMethod(listHandler.getMethod(),String.class,String.c
lass);

    String clazzName = actionClass.getName(); // 类名
    String resultStr = (String) method.invoke(object,userName,pwd); // Action
类中方法返回的结果

    // 在 ResultList 中查找对应的 name
    boolean isFind = false;

    Result result = new Result();
    for (int i = 0; i < resultList.size(); i++) {
        result = (Result) resultList.get(i);
        System.out.println("resultList 中存放的" + clazzName+ "的" +
result.getResultName());
        if
(resultStr.equals(result.getResultName())&&(result.getActionName().equals(a
ctionName.substring(1,actionName.length())))) {
            // 类的方法的返回结果与 xml 中进行比较,若找到则 break 退出,好像写
成 while 更好
            isFind = true;
            break;
        }
    }

    if (isFind) {
        // 找到了则用定义的 type 类型进行转发或重定向
        String type = result.getResultType();
        System.out.println("找到结果: " +result.getResultType() +
```

```
result.getResultValue());
        if (type.equals("redirect")) {
            response.sendRedirect(request.getContextPath()+
result.getResultValue());
        } else if(type.equals("forward")) {

request.getRequestDispatcher(result.getResultValue()).forward(request,respo
nse);
        }
    } else {
        System.out.println("未找到结果：");
        response.sendError(501,"没有请求的资源");
    }

} else  {
    // 无效的 action 返回不可识别的 action 请求
    response.sendError(500,"不可识别的 action 请求");
}
```

4.结论

1.

1.9

登陆成功(写死了,只有一种方式能成功)

登陆失败

注册(只有成功,跳回登陆界面)

重定向,url 没变

2.

对 Struct2 的理解,比较配置控制器与注解控制器的优缺点

答:

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 则分别可以分为两部分，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 WebContainer 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

对于注解控制器，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。

而配置控制器在于灵活修改，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

5.参考文献

黑马程序员 JavaWeb 就业班 Day28
XML Parser SAX: http://www.saxproject.org/quickstart.html

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