常见的设计模式：工厂，单例，代理，观察者

## 1.Strut2含义及发展历史

英文意思是：支撑杆，支持物。

发展历史：

Model 1 : JSP+JavaBean模式

Model 2: JSP+ Servlet+JavaBean 模式

Strut2之前还有一个Struts1框架，但是Struts2是继承自Webwork框架，吸收了两种的优势。

## 2.Struts2框架结构





2.在上述问题进行升级，在多URL的情况下怎么办。

步骤分析：

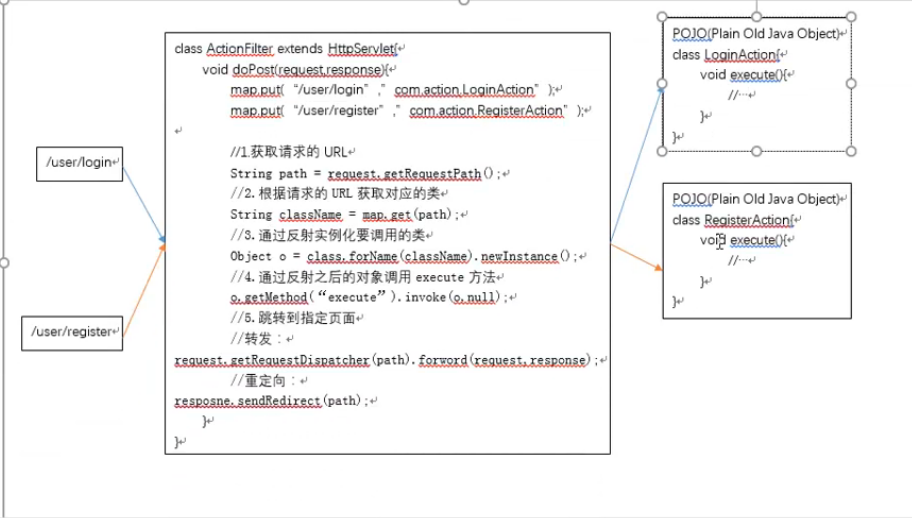
1.首先在map结构中存储url以及对应的类的全名

2.根据请求的URL获取对应的类。

3.通过反射实例化调用类。

4.通过反射之后的类调用excute方法

5.跳转到指定的页面（转发，重定向）



3.在上述问题进行升级，在多URL的情况下怎么办。

步骤分析：

1.首先在map结构中存储url以及对应的类的全名

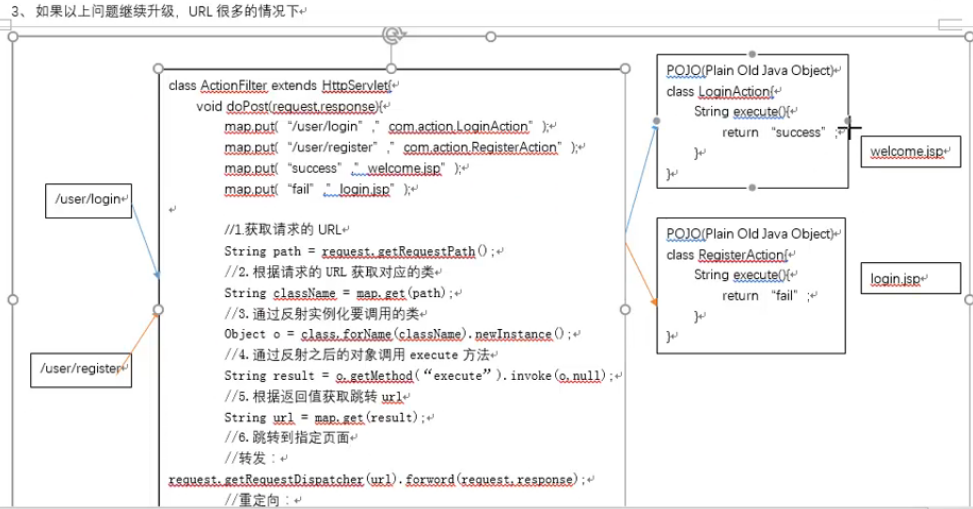
2.根据请求的URL获取对应的类。

3.通过反射实例化调用类。

4.通过反射之后的类调用excute方法

5.根据返回值回去URL

6.跳转到指定的页面（转发，重定向）



缺点：url映射，硬编码。 可以通过配置文件实现：properties, xml

<action name=”login” class=”com.action.LoginAction”>

<result name=”success”>/welcome.jsp</result>

<result name=”fail”>/login.jsp</result>

</action>

<action name=”register” class=”com.action.RegisterAction”>

<result name=”success”>/welcome.jsp</result>

<result name=”fail”>/register.jsp</result>

</action>

Struts框架引入文件配置：

Struts.xml是核心配置文件放在web/classes目录下

Struts2框架是是基于拦截器的功能实现的

1. 创建web项目
2. 添加Struts包的支持
3. 创建各种Action

第一步创建POJO类

|  |
| --- |
| package com.demo.action;  public class LoginAction {  public String execute(){  return "success";  }  } |

1. 配置struts.xml文件

|  |
| --- |
| <struts>  <constant name="struts.i18n.reload" value="false"/>  <constant name="struts.devMode" value="false"/>  <!-- <include file="struts-default.xml"/>-->  <!--  action元素详解  name：action的访问资源名称。  class：action的完整类名。  method：指定调用action类中那个方法处理请求。  -->  <package name="default" extends="struts-default" namespace="/mynamespace">  <action name="login" class="com.demo.action.DemoAction" >  <result name="success">/index.jsp</result>  <result name="error">/login.jsp</result>  </action>  </package>  </struts> |

1. 启动运行看结果

**默认的配置文件：**

1.struts-default.xml Struts默认配置文件 在struts-cored-2.2.1.jar包中

2.defult.properties 默认常量配置 org.apache.struts2

|  |  |
| --- | --- |
| struts.i18n.encoding=UTF-8 | 编码格式 |
| # struts.objectFactory = spring | SSH集成时使用 |
| struts.objectFactory.spring.autoWire = name | 自动装配 |
| struts.multipart.parser=jakarta  struts.multipart.saveDir=  struts.multipart.maxSize=2097152 | 文件上传相关配置  上传临时目录  文件最大值 |
| struts.action.extension=action,, | URL 后缀配置 可自定义  支持xxx.action和没有后缀 |
| struts.enable.DynamicMethodInvocation = false | 动态调用方法 |

3.web.xml 项目配置文件

4.struts.xml Struts核心配置文件

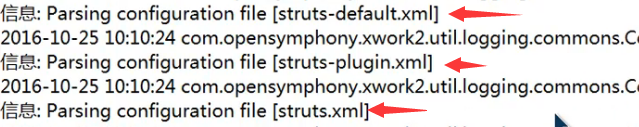
5.struts.properties 自定义常量配置文件

6.struts-plugin.xml 插件自定义

**几大配置文件优先级**

Web.xml > struts.properties > struts.xml > default.properties

**配置加载顺序**



核心配置文件

1. 常量配置

|  |
| --- |
| <constant name="struts.devMode" value="false"/> |

对default.properties中进行个性配置

1. Package元素

Action元素配置包，Java代码中的“包”，用来管理Java代码，防止命名冲突，package节点作用也是一样，防止action名冲突，在项目总指定唯一的name属性

|  |  |
| --- | --- |
| Name | 唯一标识 |
| Namespace | 命名空间，影响该包下属的action访问路径 |
| Extends | 指定继承xxx配置文件，默认继承自struts-default.xml文件 |
| Abstract | 定义为抽象包，只能别的包去继承此包，通常用于项目中的默认配置 |

1. Action元素

用于配置处理请求

结合 namespace+action<name属性>

|  |  |
| --- | --- |
| Name | Action名字，影响URL请求路径 |
| Class | 指定action全限定名 |
| method | 指定调用类方法 默认execute |
|  |  |

1. Result元素

配置action响应，可以配置多个

|  |  |
| --- | --- |
| Name | 类中方法返回值，要一致 |
| Type | 请求响应类型，默认是（转发dispacher）（可以改成重定向redirect） |

1. 协作配置文件

对于一个项目而言，通常配置 we年

|  |  |
| --- | --- |
| <include file=”zhangsan.xml”></include> | 文件路径 |

dtd文件对XML文件的约束

1. default-action-ref元素

默认action配置，通常用于防止URL输入错误，显示404错误页面，但该页面往往对用户不友好，可以通过使用该配置控制404显示页面

|  |  |
| --- | --- |
| <default-action-ref name=”default”></default-action-ref> | 当乱输URL时找不到action时,调用默认的action |

1. default-class-ref元素

Action找不到对应的Class的时候，会调用默认的Class类

|  |  |
| --- | --- |
| <default-class-ref class=”com.action.LoginAction”></default -class-ref> |  |

1. global-results元素

多个action共享的result

|  |
| --- |
| <global-results>  <result name=”error”>/500.jsp</result>  </global-results> |

**动态调用方法**

不使用action的method属性的情况下，通过URL <http://localhost:6060/struts2002/test/login!hello>

指定调用后台的xx方法 使用“action名！方法名”使用时需“struts.enable.DynamicMethodInvocation = true”设置为true，注意该配置在2.1以上的版本中默认为false .

**使用通配符**

在action的name属性中可以使用\*去命名，那么在其他属性中需要使用“{1}”或者“{2}”或者“{0}”取出URL路径。

范例：

|  |  |
| --- | --- |
| Name=”\*\_\*” | {1}取出第一个\*号  {2}取出第二个\*号 |
| url=/user\_add | {1}=user 通常出现在class属性中  {2}=add 通常出现在method属性中 |

{0}所取出来的数据就是类似如下效果：user\_add

**接受请求参数**

Struts2是MVC框架中的控制器（处理请求）那么在整个请求过程中，对参数处理就是Struts2的请求所在，接受和处理参数分为属性驱动，模型驱动，域驱动等

1. 属性驱动

直接在action的POJO类中声明与表单元素一致的属性，并生成匹配的get,set方法strut2会自动将表单数据提交到action中，并且直接给相应的属性赋值

1. 模型驱动

Struts2中提供了一个模型驱动的拦截器，要使用此模型驱动的拦截器填充参数，需要实现ModelDriven<XXX>泛型类

|  |
| --- |
| @override  Public User getModel(){  If(user==null)  User=new User();  Return user;  } |

（3）域驱动

通常情况下，表单元素往往对应一个实体，实体对象往往放在实体层中，那么需要在action中放实体、

Html代码

|  |
| --- |
|  |

Action代码LoginAction.java

|  |
| --- |
| xx |

（3）集合驱动

List<User> user

**获取ServletAPI**

ServletAPI提供什么东西？

Web容器提供各种内置对象，request,session,application等，在struts2中获取这些对象可以通过三种方式，ActionContext(Struts建议的)，ServletActionContext(可以获取真正的HttpServletRequest等)，特定接口（了解即可）

1. ActionContext

|  |
| --- |
| package com.demo.action;  import com.opensymphony.xwork2.ActionContext;  public class LoginAction {  public String execute(){  ActionContext act =ActionContext.getContext();  act.getSession().put("request","1111111111111");  act.getSession().put("session","22222222222222");  act.getSession().put("application","333333333333");  System.out.println("execute login.jsp文件");  return "success";  }  } |