## 技术分析之在Struts2框架中使用Servlet的API

### 1. 在Action类中也可以获取到Servlet一些常用的API

需求：提供JSP的表单页面的数据，在Action中使用Servlet的API接收到，然后保存到三个域对象中，最后再显示到JSP的页面上。

\* 提供JSP注册的页面，演示下面这三种方式

|  |
| --- |
| <h3>注册页面</h3>  <form action="${pageContext.request.contextPath }/xxx.action" method="post">  姓名:<input type="text" name="username" /><br/>  密码:<input type="password" name="password" /><br/>  <input type="submit" value="注册" />  </form> |

### 2. 完全解耦合的方式

如果使用该种方式，Struts2框架中提供了一个类，ActionContext类，该类中提供一些方法，通过方法获取Servlet的API

|  |
| --- |
| ActionContext是Map结构的容器,ActionContext是Action的上下文,类比ServletContext,存放着Action执行过程中的数据信息,比如Action的数据,request的数据,session的s数据,applicaton的数据等.**每次请求时会为当前线程创建一个新的ActionContext对象**.通过ThreadLocal来维护当前线程的request等数据,所以ActionContext是**线程安全**的. |

一些常用的方法如下

static ActionContext getContext() -获取ActionContext对象实例

java.util.Map<java.lang.String,java.lang.Object> getParameters() -- 获取请求参数，相当于request.getParameterMap();

java.util.Map<java.lang.String,java.lang.Object> getSession() -- 获取的代表session域的Map集合，就相当于操作session域。

java.util.Map<java.lang.String,java.lang.Object> getApplication() -- 获取代表application域的Map集合

void put(java.lang.String key, java.lang.Object value) -- 注意：向request域中存入值。

### 3. 使用原生Servlet的API的方式

Struts2框架提供了一个类，ServletActionContext，该类中提供了一些静态的方法

具体的方法如下

getPageContext()

getRequest()

getResponse()

getServletContext()

## 技术分析之结果页面的跳转

### 1. 结果页面存在两种方式

#### 全局结果页面

> 条件：如果<package>包中的一些action都返回success，并且返回的页面都是同一个JSP页面，这样就可以配置全局的结果页面。

> 全局结果页面针对的当前的包中的所有的Action，但是如果局部还有结果页面，会优先局部的。使用的标签是

<global-results>

<result>/demo3/suc.jsp</result>

</global-results>

#### 局部结果页面

<result>/demo3/suc.jsp</result>

### 2. 结果页面的类型

\* 结果页面使用<result>标签进行配置，包含两个属性

> name -- 逻辑视图的名称

> type -- 跳转的类型，值一些，需要掌握一些常用的类型。常见的结果类型去struts-default.xml中查找。

dispatcher -- 转发.type的默认值.Action--->JSP

redirect -- 重定向. Action--->JSP

chain -- 多个action之间跳转.从一个Action转发到另一个Action. Action---Action

redirectAction -- 多个action之间跳转.从一个Action重定向到另一个Action. Action---Action

stream -- 文件下载时候使用的

## 技术分析之Struts2框架的数据封装

### 1. 为什么要使用数据的封装呢？

作为MVC框架，必须要负责解析HTTP请求参数，并将其封装到Model对象中

封装数据为开发提供了很多方便

Struts2框架提供了很强大的数据封装的功能，不再需要使用Servlet的API完成手动封装了！！

### 2. Struts2中提供了两类数据封装的方式？

第一种方式：属性驱动

> 提供对应属性的set方法进行数据的封装。

表单的哪些属性需要封装数据，那么在对应的Action类中提供该属性的set方法即可。

表单中的数据提交，最终找到Action类中的setXxx的方法，最后赋值给全局变量。

注意0：Struts2的框架采用的拦截器完成数据的封装。

注意1：这种方式不是特别好:因为属性特别多,提供特别多的set方法,而且还需要手动将数据存入到对象中.

注意2：这种情况下，Action类就相当于一个JavaBean，就没有体现出MVC的思想，Action类又封装数据，又接收请求处理，耦合性较高。

> 在页面上，使用OGNL表达式进行数据封装。

\* 在页面中使用OGNL表达式进行数据的封装，就可以直接把属性封装到某一个JavaBean的对象中。

\* 在页面中定义一个JavaBean，并且提供set方法：例如：private User user:

\* 页面中的编写发生了变化，需要使用OGNL的方式，表单中的写法：<input type="text" name="user.username">

\* 注意：只提供一个set方法还不够，必须还需要提供user属性的get和set方法！！！

> 先调用get方法，判断一下是否有user对象的实例对象，如果没有，调用set方法把拦截器创建的对象注入进来，

### 第二种方式：模型驱动

> 使用模型驱动的方式，也可以把表单中的数据直接封装到一个JavaBean的对象中，并且表单的写法和之前的写法没有区别！

> 编写的页面不需要任何变化，正常编写name属性的值

> 模型驱动的编写步骤：

\* 手动实例化JavaBean，即：private User user = new User();

\* 必须实现ModelDriven<T>接口，实现getModel()的方法，在getModel()方法中返回user即可！！

## 技术分析之Struts2把数据封装到集合中

1. 封装复杂类型的参数（集合类型 Collection 、Map接口等）

2. 需求：页面中有可能想批量添加一些数据，那么现在就可以使用上述的技术了。把数据封装到集合中

3. 把数据封装到Collection中

因为Collection接口都会有下标值，所有页面的写法会有一些区别，注意：

> <input type="text" name="products[0].name" />

在Action中的写法，需要提供products的集合，并且提供get和set方法。

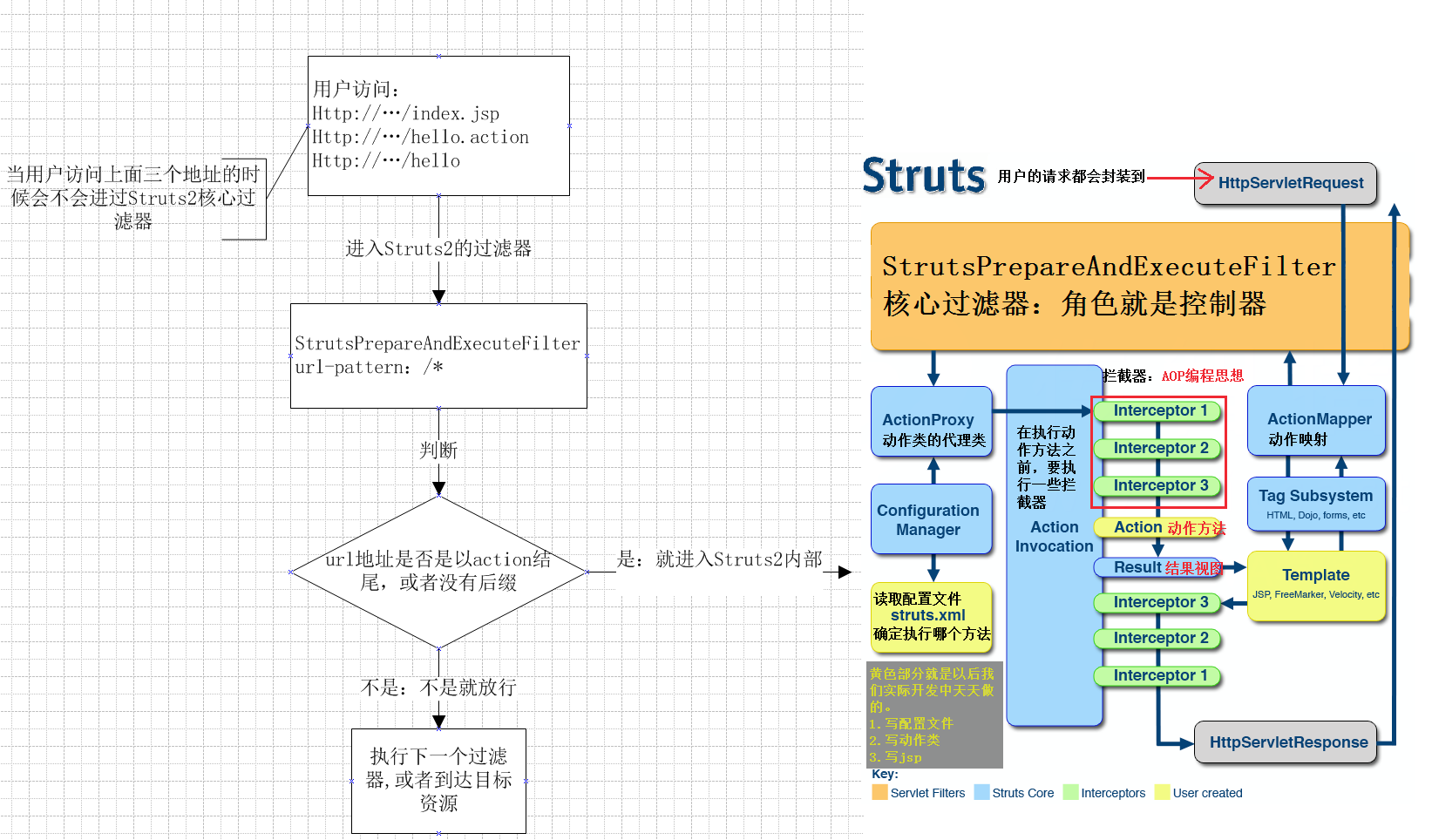
4. 把数据封装到Map中

Map集合是键值对的形式，页面的写法

> <input type="text" name="map['one'].name" />

Action中提供map集合，并且提供get和set方法

## 案例总结之Struts2的拦截器技术

****

com.opensymphony.xwork2.interceptor.Interceptor 拦截器的接口

import com.opensymphony.xwork2.interceptor.MethodFilterInterceptor，指定来不过滤哪些方法。

### 1. 拦截器的概述

拦截器就是AOP（Aspect-Oriented Programming）的一种实现。（AOP是指用于在某个方法或字段被访问之前，进行拦截然后在之前或之后加入某些操作。）

过滤器:过滤从客服端发送到服务器端请求的（可以过滤jsp,js,servlet,css,img）

拦截器:拦截对目标Action中的某些方法进行拦截

拦截器不能拦截JSP

拦截到Action中某些方法

### 2. 拦截器和过滤器的区别

1）拦截器是基于JAVA反射机制的，而过滤器是基于函数回调的

2）过滤器依赖于Servlet容器，而拦截器不依赖于Servlet容器

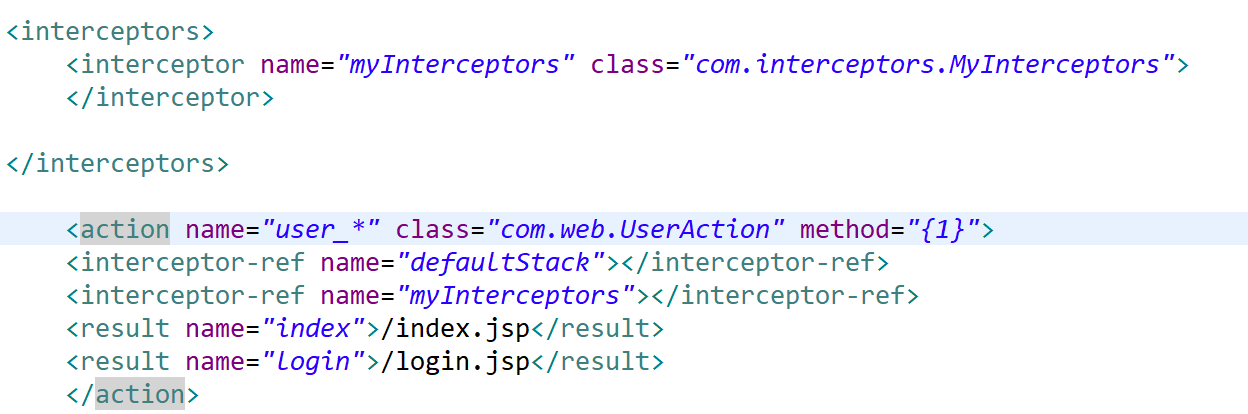
3）拦截器只能对Action请求起作用（Action中的方法），而过滤器可以对几乎所有的请求起作用（CSS JSP JS）

拦截器 采用 责任链 模式

> 在责任链模式里,很多对象由每一个对象对其下家的引用而连接起来形成一条链

> 责任链每一个节点，都可以继续调用下一个节点，也可以阻止流程继续执行

在struts2 中可以定义很多个拦截器，将多个拦截器按照特定顺序 组成拦截器栈 （顺序调用 栈中的每一个拦截器 ）



### 3. Struts2的核心是拦截器，看一下Struts2的运行流程

案例总结之自定义拦截器和配置

1. 编写拦截器，需要实现Interceptor接口，实现接口中的三个方法

protected String doIntercept(ActionInvocation invocation) throws Exception {

// 获取session对象

User user = (User) ServletActionContext.getRequest().getSession().getAttribute("existUser");

if(user == null){

// 说明，没有登录，后面就不会执行了

return "login";

}

return invocation.invoke();

}

2. 需要在struts.xml中进行拦截器的配置，配置一共有两种方式

<!-- 定义了拦截器 第一种方式

<interceptors>

<interceptor name="DemoInterceptor" class="com.qf.interceptor.DemoInterceptor"/>

</interceptors>

-->

<interceptors>

<interceptor name="DemoInterceptor" class="com.qf.interceptor.DemoInterceptor"/>

<!-- 定义拦截器栈 -->

<interceptor-stack name="myStack">

<interceptor-ref name="DemoInterceptor"/>

<interceptor-ref name="defaultStack"/>

</interceptor-stack>

</interceptors>

<action name="userAction" class="com.qf.demo3.UserAction">

<!-- 只要是引用自己的拦截器，默认栈的拦截器就不执行了，必须要手动引入默认栈

<interceptor-ref name="DemoInterceptor"/>

<interceptor-ref name="defaultStack"/>

-->

<!-- 引入拦截器栈就OK -->

<interceptor-ref name="myStack"/>

</action>

## jquery validate校验表单数据

## 案例

### 注册用户

### 使用拦截器判断用户是否已经登录