自定义拦截器

自定义拦截器

- Struts2中的拦截器类似于JAVAEE中的过滤器，但是不同的是过滤器Filter可以拦截过滤所有的请求，但是拦截器仅限于拦截所有的Action请求

- 创建一个自定义拦截器的方式：
  - 实现Interceptor接口
  - 覆盖init,destroy,intercept三个抽象方法

- 拦截器的方法：
  - init方法会在拦截器对象被初始化启动时执行
  - destroy方法会在拦截器对象被销毁时执行
  - intercept方法会在请求指定Action时执行

拦截器代码示例

```
package intercept;
import javax.servlet.http.HttpSession;
import org.apache.struts2.ServletActionContext;
import com.opensymphony.xwork2.ActionInvocation;
import com.opensymphony.xwork2.interceptor.Interceptor;

public class CheckLoginIntercept implements Interceptor {

	public void init() {
		System.out.println("拦截器初始化");
	}

	public void destroy() {
		System.out.println("拦截器被销毁");
	}

	public String intercept(ActionInvocation invocation) throws Exception {
		HttpSession session = ServletActionContext.getRequest().getSession();

		if (session.getAttribute("username") != null) {
			System.out.println("拦截器验证登陆成功");
			return invocation.invoke();//继续向下执行
		} else {
			System.out.println("拦截器验证登陆失败");
			return "login.jsp";
		}
	}
}
```

拦截器配置

- 在package标签中配置一个<interceptors>标签，在<interceptors>中配置一个到多个<interceptor>，每个<interceptor>都代表一个拦截器，然后在需要请求之前执行拦截器的Action标签中加入引用拦截器的<interceptor-ref>标签，但是千万一定不要忘记在引用自定义拦截器时还要引用默认拦截器栈defaultStack，否则会丧失struts2的基本功能

拦截器配置示例

```
<struts>
    <package name="vip" namespace="/vip" extends="struts-default">
        <!-- 包中的拦截器声明 -->
        <interceptors>
            <interceptor name="checkLogin" class="intercept.CheckLoginIntercept"/>
        </interceptors>

        <action name="vipAction" class="action.VipAction" method="execute">

            <!--在Action中引用拦截器，但一定要在最前面引用默认拦截器栈-->
            <interceptor-ref name="defaultStack"/>
            <interceptor-ref name="checkLogin"/>


            <result name="index.jsp">/index.jsp</result>
            <result name="login.jsp">/login.jsp</result>
        </action>
    </package>
</struts>
```

拦截器栈

- 拦截器栈实际上就是配置一组拦截器,当Action引用了某个拦截器栈的时候就相当于引用了这一组拦截器,拦截器执行的顺序以配置的顺序为准
- 拦截器栈需要用到<interceptor-stack>标签

拦截器栈配置示例

```
<struts>
    <package name="vip" namespace="/vip" extends="struts-default">
        <!-- 包中的拦截器声明 -->
        <interceptors>
            <interceptor name="checkLogin" class="intercept.CheckLoginIntercept"/>

            <!-- 拦截器栈声明 -->
            <interceptor-stack name="checkLoginStack">
                <interceptor-ref name="defaultStack"/>
                <interceptor-ref name="checkLogin"/>

            </interceptor-stack>
        </interceptors>

        <action name="*" class="action.VipAction" method="{1}">
            <!--在Action中引用拦截器栈-->
            <interceptor-ref name="checkLoginStack"/>

            <result name="index.jsp">/index.jsp</result>
            <result name="login.jsp">/login.jsp</result>
        </action>

    </package>
</struts>
```

默认拦截器

- 如果一个包中有很多的Action，而这些Action都想执行相同的拦截器，可以使用<default-interceptor-ref>来调用默认拦截器，但是需要注意的是如果Action中显式的引用了任何拦截器，则默认拦截器不起作用

默认拦截器代码示例

```
<struts>
    <package name="vip" namespace="/vip" extends="struts-default">
        <!-- 包中的拦截器声明 -->
        <interceptors>
            <interceptor name="checkLogin" class="intercept.CheckLoginIntercept"/>

            <!-- 拦截器栈声明 -->
            <interceptor-stack name="checkLoginStack">

                <interceptor-ref name="defaultStack"/>
                <interceptor-ref name="checkLogin"/>

            </interceptor-stack>
        </interceptors>

        <!-- 让包中的所有Action都默认应用此拦截器 -->
        <default-interceptor-ref name="checkLoginStack"/>

        <action name="*" class="action.VipAction" method="{1}">

            <result name="index.jsp">/index.jsp</result>
            <result name="login.jsp">/login.jsp</result>
        </action>
    </package>
</struts>
```