**spring mvc的拦截器**

SpringMVC 中的Interceptor 拦截器也是相当重要和相当有用的，它的主要作用是拦截用户的请求并进行相应的处理。比如通过它来进行**权限验证**，或者是来***判断用户是否登陆***，或者是像12306 那样子***判断当前时间是否是购票时间。***

**一、定义Interceptor实现类**

SpringMVC 中的Interceptor 拦截请求是通过***HandlerInterceptor*** 来实现的。在SpringMVC 中定义一个Interceptor 非常简单，主要有两种方式，

第一种方式是要定义的Interceptor类要实现了Spring 的***HandlerInterceptor***接口，或者是这个类继承实现了HandlerInterceptor 接口的类，比如Spring 已经提供的实现了HandlerInterceptor 接口的抽象类HandlerInterceptorAdapter ；

第二种方式是实现Spring的***WebRequestInterceptor***接口，或者是继承实现了WebRequestInterceptor的类。

**（一）实现HandlerInterceptor接口**

HandlerInterceptor 接口中定义了三个方法，我们就是通过这三个方法来对用户的请求进行拦截处理的。

（1 ）preHandle (HttpServletRequest request, HttpServletResponse response, Object handle) 方法，顾名思义，该方法将在请求处理之前进行调用。SpringMVC 中的Interceptor 是链式的调用的，在一个应用中或者说是在一个请求中可以同时存在多个Interceptor 。每个Interceptor 的调用会依据它的声明顺序依次执行，而且最先执行的都是Interceptor 中的preHandle 方法，所以可以在这个方法中进行一些前置初始化操作或者是对当前请求的一个预处理，也可以在这个方法中进行一些判断来决定请求是否要继续进行下去。该方法的返回值是布尔值Boolean 类型的，当它返回为false 时，表示请求结束，后续的Interceptor 和Controller 都不会再执行；当返回值为true 时就会继续调用下一个Interceptor 的preHandle 方法，如果已经是最后一个Interceptor 的时候就会是调用当前请求的Controller 方法。

（2 ）postHandle (HttpServletRequest request, HttpServletResponse response, Object handle, ModelAndView modelAndView) 方法，由preHandle 方法的解释我们知道这个方法包括后面要说到的afterCompletion 方法都只能是在当前所属的Interceptor 的preHandle 方法的返回值为true 时才能被调用。postHandle 方法，顾名思义就是在当前请求进行处理之后，也就是Controller 方法调用之后执行，但是它会在DispatcherServlet 进行视图返回渲染之前被调用，所以我们可以在这个方法中对Controller 处理之后的ModelAndView 对象进行操作。postHandle 方法被调用的方向跟preHandle 是相反的，也就是说先声明的Interceptor 的postHandle 方法反而会后执行，这和Struts2 里面的Interceptor 的执行过程有点类型。Struts2 里面的Interceptor 的执行过程也是链式的，只是在Struts2 里面需要手动调用ActionInvocation 的invoke 方法来触发对下一个Interceptor 或者是Action 的调用，然后每一个Interceptor 中在invoke 方法调用之前的内容都是按照声明顺序执行的，而invoke 方法之后的内容就是反向的。

（3 ）afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handle, Exception ex) 方法，该方法也是需要当前对应的Interceptor 的preHandle 方法的返回值为true 时才会执行。顾名思义，该方法将在整个请求结束之后，也就是在DispatcherServlet 渲染了对应的视图之后执行。这个方法的主要作用是用于进行资源清理工作的。

配置具体步骤：

一、自定义一个拦截器类

[复制代码](javascript:void(0);)

package cn.yxj.MyIntercpetor;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.springframework.web.servlet.HandlerInterceptor;

import org.springframework.web.servlet.ModelAndView;

public class MyIntercpetor implements HandlerInterceptor {

//请求执行方法之前执行

@Override

public boolean preHandle(HttpServletRequest request,

HttpServletResponse response, Object handler) throws Exception {

System.out.println("MyIntercpetor.preHandle()");

return true;

}

//执行方法之后执行

@Override

public void postHandle(HttpServletRequest request,

HttpServletResponse response, Object handler,

ModelAndView modelAndView) throws Exception {

System.out.println("MyIntercpetor.postHandle()");

}

//最终执行的方法

@Override

public void afterCompletion(HttpServletRequest request,

HttpServletResponse response, Object handler, Exception ex)

throws Exception {

System.out.println("MyIntercpetor.afterCompletion()");

}

}

[复制代码](javascript:void(0);)

二、自定义一个处理器

[复制代码](javascript:void(0);)

package cn.yxj.controller;

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.annotation.RequestMapping;

@Controller

public class MyController {

// 处理器方法

@RequestMapping(value = "/first.do")

public String doFirst() {

// 所有文件上传

System.out.println("MyController.doFirst()");

return "welcome.jsp";

}

}

[复制代码](javascript:void(0);)

三、配置文件

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>

<beans xmlns="http://www.springframework.org/schema/beans"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns:context="http://www.springframework.org/schema/context"

xmlns:aop="http://www.springframework.org/schema/aop"

xmlns:tx="http://www.springframework.org/schema/tx"

xmlns:mvc="http://www.springframework.org/schema/mvc"

xsi:schemaLocation="

http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd

http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context.xsd

http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop.xsd

http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx.xsd

http://www.springframework.org/schema/mvc http://www.springframework.org/schema/mvc/spring-mvc.xsd

">

<!-- 包扫描器 -->

<context:component-scan base-package="cn.yxj.controller"></context:component-scan>

<!-- 配置拦截器 -->

<mvc:interceptors>

<mvc:interceptor>

<mvc:mapping path="/\*\*"/>

<bean class="cn.yxj.MyIntercpetor.MyIntercpetor"/>

</mvc:interceptor>

</mvc:interceptors>

<!-- 配置拦截器多个 -->  
　　　　<mvc:interceptors>  
　　　　　　<mvc:interceptor>  
　　　　　　　　<mvc:mapping path="/\*\*"/>  
　　　　　　　　　　<bean class="cn.yxj.MyIntercpetor.SendceInterceptor"/>  
　　　　　　　　</mvc:interceptor>  
　　　　　　　</mvc:interceptors>

</beans>

[复制代码](javascript:void(0);)

四、页面就是普通页面

五、在web.xml中配置url节点的\*.do

[复制代码](javascript:void(0);)

<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://java.sun.com/xml/ns/javaee" xmlns:web="http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" version="2.5">

<display-name></display-name>

<servlet>

<servlet-name>springmvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<init-param>

<param-name>contextConfigLocation</param-name>

<param-value>classpath:applicationContext.xml</param-value>

</init-param>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>springmvc</servlet-name>

<url-pattern>\*.do</url-pattern>

</servlet-mapping>

<welcome-file-list>

<welcome-file>index.jsp</welcome-file>

</welcome-file-list>

</web-app>

[复制代码](javascript:void(0);)

多个拦截器就在配置文件中配置，在多加一个拦截器类。

六、解释/\*\*



 七拦截器的执行流程



多个拦截器的执行流程

