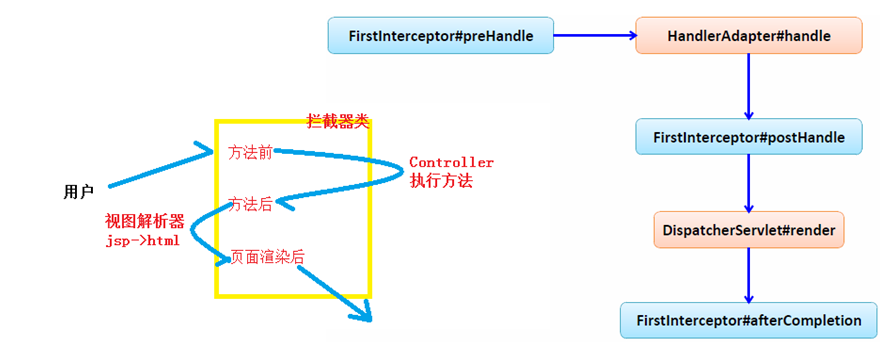
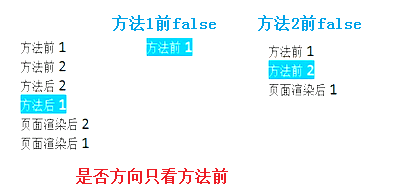
# 概述

## 定义

Spring Web MVC 的处理器拦截器类似于Servlet 开发中的过滤器Filter，用于对处理器进行预处理和后处理。

## 单个执行顺序





总结：

preHandle按拦截器定义顺序调用

postHandler按拦截器定义逆序调用；在拦截器链内所有拦截器返成功调用

afterCompletion按拦截器定义逆序调用；只有preHandle返回true才调用

## 拦截器方法定义

**preHandle**()：这个方法在业务处理器处理请求之前被调用，在该方法中对用户请求 request 进行处理。如果程序员决定该拦截器对请求进行拦截处理后还要调用其他的拦截器，或者是业务处理器去进行处理，则返回**true**；如果程序员决定不需要再调用其他的组件去处理请求，则返回**false**。

**postHandle**()：这个方法在业务处理器处理完请求后，但是**DispatcherServlet** 向客户端返回响应前被调用，在该方法中对用户请求request进行处理。

**afterCompletion**()：这个方法在 **DispatcherServlet** 完全处理完请求后被调用，可以在该方法中进行一些资源清理的操作。

实现HandlerInterceptor接口，如下：



## 拦截器配置

上面定义的拦截器再复制一份HandlerInterceptor2，注意新的拦截器修改代码：

System.***out***.println("HandlerInterceptor2....preHandle");

在springmvc.xml中配置拦截器

<!-- 配置拦截器 -->

<mvc:interceptors>

<mvc:interceptor>

<!-- 所有的请求都进入拦截器 -->

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

<!-- 配置具体的拦截器 -->

<bean class=*"cn.itcast.ssm.interceptor.HandlerInterceptor1"* />

</mvc:interceptor>

<mvc:interceptor>

<!-- 所有的请求都进入拦截器 -->

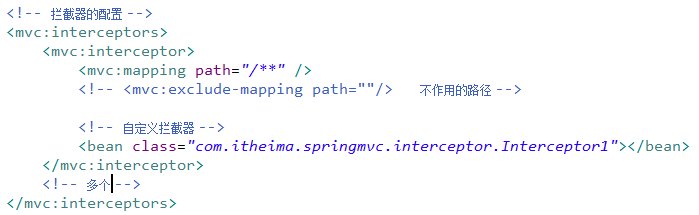
<mvc:mapping path=*"/\*\*"* />

<!-- 配置具体的拦截器 -->

<bean class=*"cn.itcast.ssm.interceptor.HandlerInterceptor2"* />

</mvc:interceptor>

</mvc:interceptors>

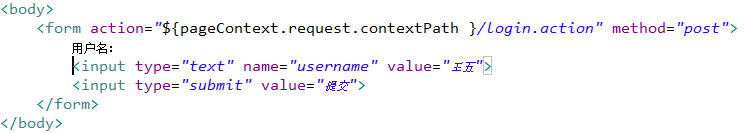


# 拦截器应用

## 处理流程

1. 有一个登录页面，需要写一个Controller访问登录页面
2. 登录页面有一提交表单的动作。需要在Controller中处理。
   1. 判断用户名密码是否正确（在控制台打印）
   2. 如果正确,向session中写入用户信息（写入用户名username）
   3. 跳转到商品列表
3. 拦截器。
   1. 拦截用户请求，判断用户是否登录（登录请求不能拦截）
   2. 如果用户已经登录。放行
   3. 如果用户未登录，跳转到登录页面。

## 编写登录login.jsp



## 用户登陆Controller

@Controller

@RequestMapping("user")

**public** **class** UserController {

/\*\*

\* 跳转到登录页面

\*

\* **@return**

\*/

@RequestMapping("toLogin")

**public** String toLogin() {

**return** "login";

}

/\*\*

\* 用户登录

\*

\* **@param** username

\* **@param** password

\* **@param** session

\* **@return**

\*/

@RequestMapping("login")

**public** String login(String username, String password, HttpSession session) {

// 校验用户登录

System.***out***.println(username);

System.***out***.println(password);

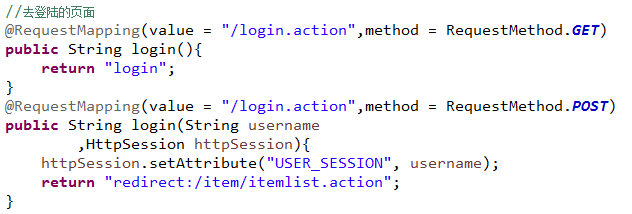
// 把用户名放到session中

session.setAttribute("username", username);

**return** "redirect:/item/itemList.action";

}

}



## 编写拦截器

@Override

**public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object arg2) **throws** Exception {

// 从request中获取session

HttpSession session = request.getSession();

// 从session中获取username

Object username = session.getAttribute("username");

// 判断username是否为null

**if** (username != **null**) {

// 如果不为空则放行

**return** **true**;

} **else** {

// 如果为空则跳转到登录页面

response.sendRedirect(request.getContextPath() + "/user/toLogin.action");

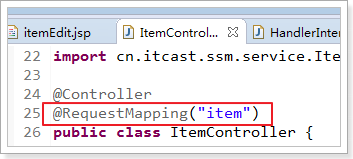
}

**return** **false**;

}

## 配置拦截器

只能拦截商品的url，所以需要修改ItemController，让所有的请求都必须以item开头，如下图：



在springmvc.xml配置拦截器

<mvc:interceptor>

<!-- 配置商品被拦截器拦截 -->

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

<!-- 配置具体的拦截器 -->

<bean class=*"cn.itcast.ssm.interceptor.LoginHandlerInterceptor"* />

</mvc:interceptor>

# 使用“/”处理静态资源

## 为什么会有这样的问题:

优雅的 REST 风格的资源URL 不希望带 .action 或 .do 等后缀

若将 DispatcherServlet 请求映射配置为 /,

则 Spring MVC 将捕获 WEB 容器的所有请求, 包括静态资源的请求, SpringMVC 会将他们当成一个普通请求处理,

因找不到对应处理器将导致错误。

<servlet-mapping>的<url-pattern>/</url-pattern>把所有的请求都交给spring去处理了，而所有available的请求url都是在Constroller里使用类似@RequestMapping(value = "/login/{user}", method = RequestMethod.GET)这样的注解配置的，这样的话对js/css/jpg/gif等静态资源的访问就会得不到。

## 解决方法1

在 SpringMVC 的配置文件中配置 <mvc:default-servlet-handler/>

<!-- 解决静态资源

将在 SpringMVC 上下文中定义一个DefaultServletHttpRequestHandler，

它会对进入 DispatcherServlet 的请求进行筛查，如果发现是没有经过映射的请求，

就将该请求交由 WEB 应用服务器默认的 Servlet 处理，如果不是静态资源的请求，

才由DispatcherServlet 继续处理

一般 WEB 应用服务器默认的 Servlet 的名称都是 default。若所使用的WEB 服务器

的默认 Servlet 名称不是 default，则需要通过 default-servlet-name 属性显式指定

-->

<mvc:default-servlet-handler/>

<mvc:annotation-driven />

## 解决方法2



## 解决方法3

在web.xml里添加如下配置

<servlet-mapping>

<servlet-name>default</servlet-name>

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

</servlet-mapping>

<servlet-mapping>

<servlet-name>default</servlet-name>

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

</servlet-mapping>

<servlet-mapping>

<servlet-name>default</servlet-name>

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

</servlet-mapping>

要配置多个，每种文件配置一个。  
要写在DispatcherServlet的前面， 让defaultServlet先拦截，这个就不会进入Spring了，我想性能是最好的吧。

Tomcat, Jetty, JBoss, and GlassFish  默认 Servlet的名字 -- "default"  
Google App Engine 默认 Servlet的名字 -- "\_ah\_default"  
Resin 默认 Servlet的名字 -- "resin-file"  
WebLogic 默认 Servlet的名字  -- "FileServlet"  
WebSphere  默认 Servlet的名字 -- "SimpleFileServlet"