



黑马程序员™
www.itheima.com

传智播客旗下
高端IT教育品牌

SpringMVC拦截器

■ 1. SpringMVC拦截器

1.1 拦截器 (interceptor) 的作用

Spring MVC 的**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**。

将拦截器按一定的顺序联结成一条链，这条链称为**拦截器链 (Interceptor Chain)**。在访问被拦截的方法或字段时，拦截器链中的拦截器就会按其之前定义的顺序被调用。拦截器也是AOP思想的具体实现。

1. SpringMVC拦截器

1.2 拦截器和过滤器区别

区别	过滤器 (Filter)	拦截器 (Interceptor)
使用范围	是 servlet 规范中的一部分，任何 Java Web 工程都可以使用	是 SpringMVC 框架自己的，只有使用了 SpringMVC 框架的工程才能用
拦截范围	在 url-pattern 中配置了/*之后，可以对所有要访问的资源拦截	在<mvc:mapping path= "" />中配置了/*之后，也可以多所有资源进行拦截，但是可以通过<mvc:exclude-mapping path= "" />标签排除不需要拦截的资源

1. SpringMVC拦截器

1.3 拦截器是快速入门


自定义拦截器很简单，只有如下三步：

- ① 创建拦截器类实现HandlerInterceptor接口
- ② 配置拦截器
- ③ 测试拦截器的拦截效果

1. SpringMVC拦截器

1.3 拦截器是快速入门

① 创建拦截器类实现HandlerInterceptor接口

```
public class MyHandlerInterceptor1 implements HandlerInterceptor {  
    public boolean preHandle(HttpServletRequest request, HttpServletResponse  
        response, Object handler) {  
        System.out.println("preHandle running...");  
        return true;  return true代表放行，false代表不放行  
    }  
  
    public void postHandle(HttpServletRequest request, HttpServletResponse  
        response, Object handler, ModelAndView modelAndView) {  
        System.out.println("postHandle running...");  
    }  
  
    public void afterCompletion(HttpServletRequest request, HttpServletResponse  
        response, Object handler, Exception ex) {  
        System.out.println("afterCompletion running...");  
    }  
}
```

1. SpringMVC拦截器

1.3 拦截器是快速入门

② 配置拦截器

```
<!--配置拦截器-->  
<mvc:interceptors>  
    <mvc:interceptor>  
        <mvc:mapping path="/**" />  
        <bean class="com.ithema.interceptor.MyHandlerInterceptor1" />  
    </mvc:interceptor>  
</mvc:interceptors>
```

 拦截所有访问

1. SpringMVC拦截器

1.3 拦截器是快速入门

③ 测试拦截器的拦截效果（编写目标方法）

```
@RequestMapping("/quick23")
@ResponseBody
public ModelAndView quickMethod23() throws IOException, ParseException {
    System.out.println("目标方法执行....");
    ModelAndView modelAndView = new ModelAndView();
    modelAndView.addObject("name", "itcast");
    modelAndView.setViewName("index");
    return modelAndView;
}
```

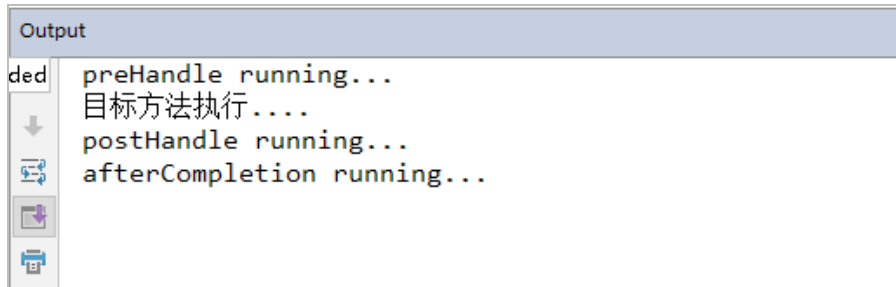
1. SpringMVC拦截器

1.3 拦截器是快速入门

③ 测试拦截器的拦截效果（访问网址）

```
http://localhost:8080/itheima_springmvc1/quick23
```

控制台打印结果



The screenshot shows an IDE's Output window with a title bar labeled "Output". On the left side of the window is a vertical toolbar containing icons for "ded" (likely "deduplicate"), a downward arrow, a refresh/clear icon, a plus icon for adding filters, and a printer icon. The main text area of the Output window displays the following log messages in a monospaced font:

```
preHandle running...  
目标方法执行....  
postHandle running...  
afterCompletion running...
```


1. SpringMVC拦截器

1.4 多拦截器操作

同上，在编写一个MyHandlerInterceptor2操作，测试执行顺序

```
↑ preHandle running...  
↓ preHandle running222...  
目标方法执行....  
postHandle running222...  
postHandle running...  
afterCompletion running222...  
afterCompletion running...
```

1. SpringMVC拦截器

1.5 拦截器方法说明

preHandle() → 目标函数的modelAndView() → postHandle() → afterCompletion()

在这里的方法中可
以对ModelAndView

方法名	说明
preHandle()	方法将在请求处理之前进行调用，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
postHandle()	该方法是在当前请求进行处理之后被调用，前提是preHandle 方法的返回值为true 时才能被调用，且它会在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
afterCompletion()	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，前提是preHandle 方法的返回值为true 时才能被调用

1. SpringMVC拦截器

1.6 知识要点

自定义拦截器步骤

- ① 创建拦截器类实现HandlerInterceptor接口
- ② 配置拦截器
- ③ 测试拦截器的拦截效果

实现prehandle(), posthandle()
, aftercompletion()方法, 主
要是prehandle方法

在spring-mvc.xml 中配置

```
<!-- 配置interceptor拦截器 -->
<mvc:interceptors>
  <mvc:interceptor>
    <!-- 配置对哪些资源进行拦截操作 -->
    <mvc:mapping path="/**"/>
    <!-- 配置对哪些资源不进行拦截操作 -->
    <mvc:exclude-mapping path="/user/login"/>
    <bean class="com.ithema.interceptor.MyInterceptor"></bean>
  </mvc:interceptor>
</mvc:interceptors>
```

1. SpringMVC拦截器

1.7 案例-用户登录权限控制

需求：用户没有登录的情况下，不能对后台菜单进行访问操作，点击菜单跳转到登录页面，只有用户登录成功后才能进行后台功能的操作

The diagram illustrates the user login and role management process in the ITCAST system. It consists of three main components:

- ITCAST后台管理系统 (ITCAST Backend Management System):** The login page shows a username field (zhangsan) and a password field (masked with dots). A red box highlights these fields, and a red arrow points from the password field to the '角色管理' (Role Management) menu item in the sidebar.
- 数据后台管理 (Data Backend Management):** The sidebar menu shows the user is logged in (在线). The '角色管理' (Role Management) menu item is highlighted with a red box, and a red arrow points from it to the '列表' (List) table.
- 列表 (List):** A table showing the roles in the system. The table has columns for ID, 角色名称 (Role Name), and 角色描述 (Role Description). The table is highlighted with a red border.

ID	角色名称	角色描述
1	院长	负责全面工作
2	研究员	课程研发工作
3	讲师	授课工作
4	助教	协助解决学生的问题
5	班主任	负责学生的生活
7	就业指导	负责学生的就业工作



传智播客旗下高端IT教育品牌