

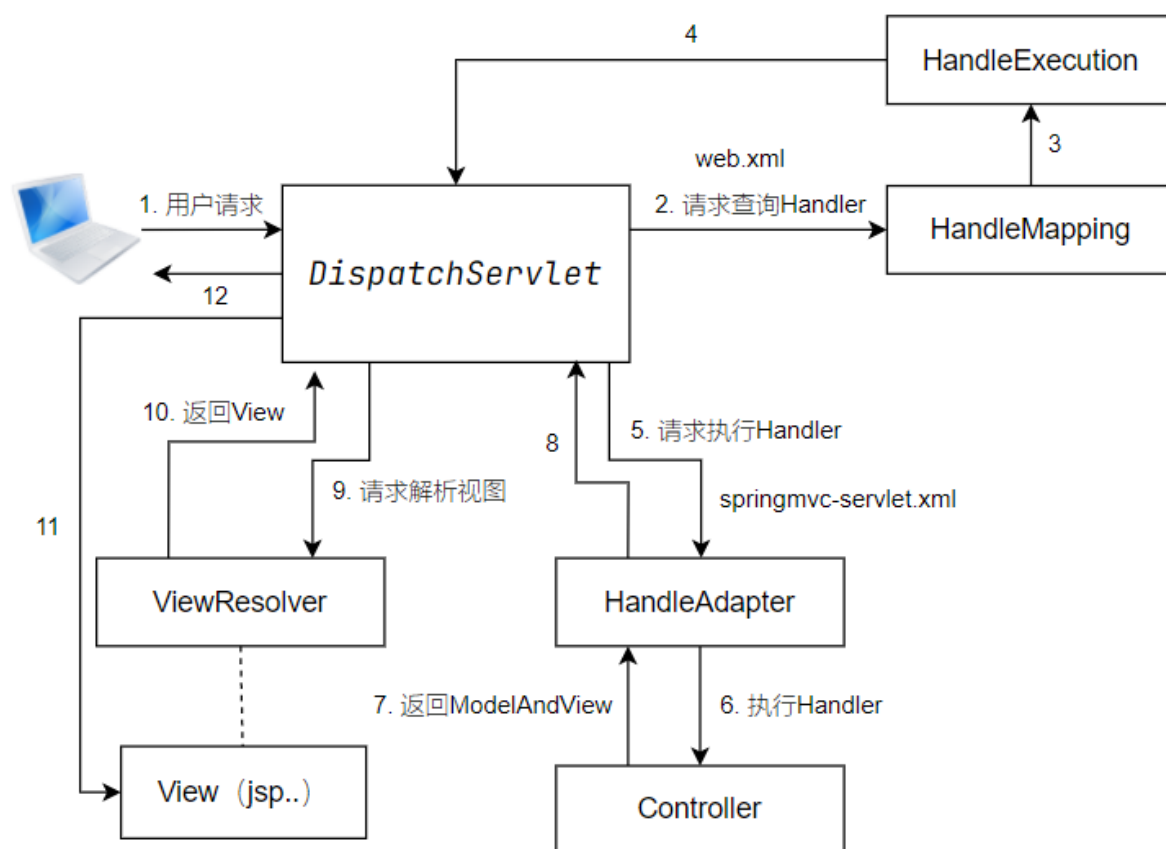
SpringMVC学习笔记

资源

官方文档: [Web on Servlet Stack \(spring.io\)](http://spring.io)

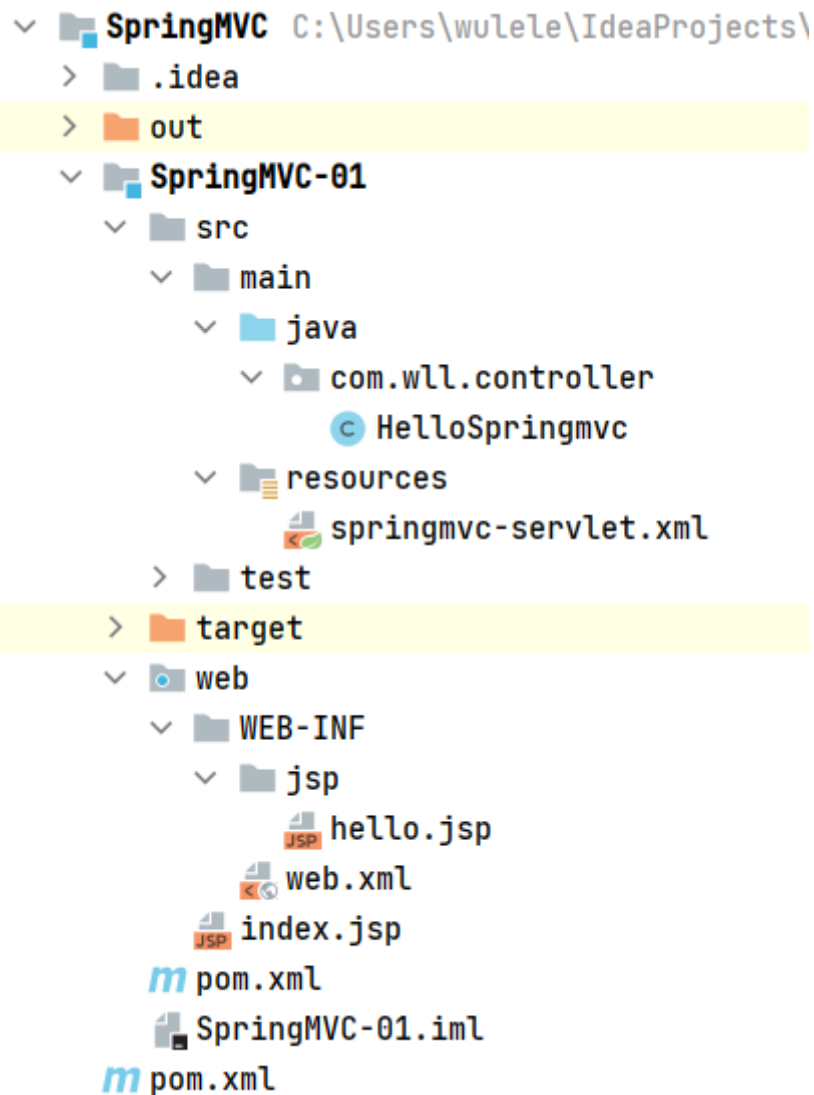
中文文档: [Spring Framework 4.3.21.RELEASE 中文文档 - 22. Web MVC 框架](#) | Docs4dev

SpringMVC工作流程（重点）



HelloSpringMVC

1. 文件结构



2. pom.xml

```
<dependencies>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>servlet-api</artifactId>
    <version>2.5</version>
  </dependency>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.13</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet</groupId>
    <artifactId>jstl</artifactId>
    <version>1.2</version>
  </dependency>
  <dependency>
    <groupId>javax.servlet.jsp</groupId>
    <artifactId>jsp-api</artifactId>
    <version>2.1</version>
  </dependency>
</dependencies>
```

```

</dependency>
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-webmvc</artifactId>
    <version>5.3.6</version>
</dependency>
</dependencies>

```

3. web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd"
    version="4.0">
    <!-- 注册DispatchServlet -->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</serv
let-class>
        <!-- 关联springmvc配置文件 -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:springmvc-
servlet.xml</param-value>
        </init-param>
        <!-- 设置启动级别为1级 -->
        <load-on-startup>1</load-on-startup>
    </servlet>
    <!-- / 匹配所有请求，浏览器地址栏，不包括.jsp -->
    <!-- /* 匹配所有请求，包括.jsp -->
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>

```

4. springmvc-servlet.xml

处理器映射器、处理器适配器、视图解析器

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

```

```

xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">

    <!-- 注册处理器映射器 -->
    <bean
class="org.springframework.web.servlet.handler.BeanNameUrlHan
dlerMapping"/>
    <!-- 注册处理器适配器 -->
    <bean
class="org.springframework.web.servlet.mvc.SimpleControllerHa
ndlerAdapter"/>
    <!-- 视图解析器 -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceV
iewResolver" id="internalResourceViewResolver">
        <!-- 前缀 -->
        <property name="prefix" value="/WEB-INF/jsp/" />
        <!-- 后缀 -->
        <property name="suffix" value=".jsp" />
    </bean>

    <!-- BeanNameUrlHandlerMapping: bean 处理/hello 请求跳转到
Controller -->
    <bean class="com.wll.controller.HelloSpringmvc"
id="/hello"/>
</beans>

```

5. HelloSpringmvc

```

/**
 * @author wulele
 */
public class HelloSpringmvc implements Controller {
    @Override
    public ModelAndView handleRequest(HttpServletRequest request,
        HttpServletResponse response) throws Exception {
        //ModelAndView 模型和视图
        ModelAndView view = new ModelAndView();

        // 业务层

        // 封装对象
        view.addObject("msg", "Hello SpringMVC");
        // /WEB-INF/jsp/hello.jsp
    }
}

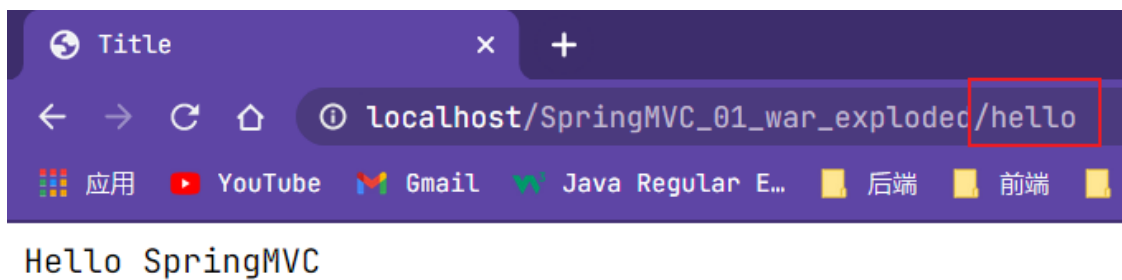
```

```
        view.setViewName("hello");  
        return view;  
    }  
}
```

6. hello.jsp

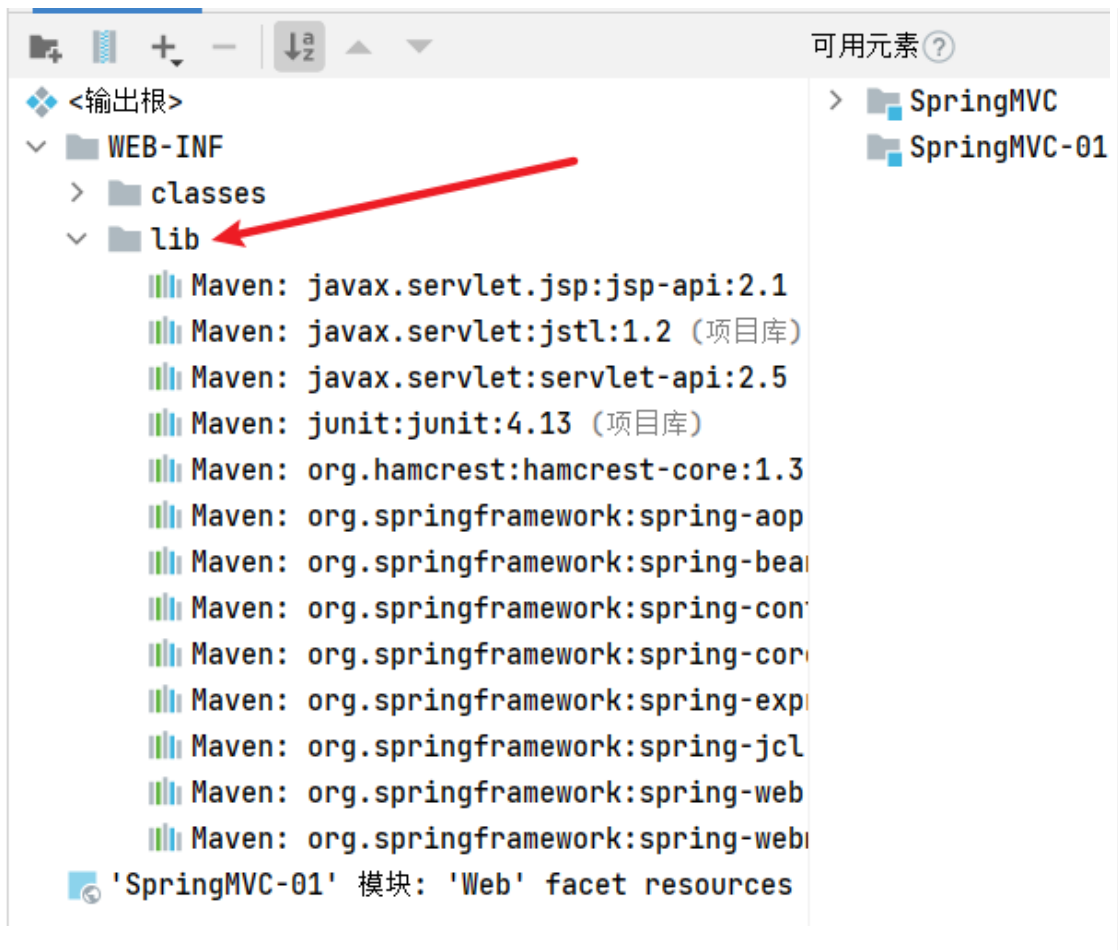
```
<%@ page contentType="text/html; charset=UTF-8"  
    language="java" %>  
<html>  
    <head>  
        <title>Title</title>  
    </head>  
    <body>  
        ${msg}  
    </body>  
</html>
```

7. 运行测试



8. 注意点

如果页面404，打开idea项目配置，在工件（artifact）选项添加lib目录添加jar包



使用注解

1. springmvc-servlet.xml

```
<?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:mvc="http://www.springframework.org/schema/mvc"

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

       http://www.springframework.org/schema/mvc/spring-mvc.xsd">
    <!-- 包扫描 -->
```

```

<context:component-scan base-
package="com.wll.controller"/>
    <!-- 将静态资源转交servlet处理，如果不是静态资源则由
DispatcherServlet处理-->
    <mvc:default-servlet-handler/>
    <!-- 注解驱动开启-->
    <mvc:annotation-driven/>

    <bean
class="org.springframework.web.servlet.view.InternalResourceV
iewResolver" id="internalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/" />
        <property name="suffix" value=".jsp" />
    </bean>
</beans>

```

2. HelloSpringmvc

```

/**
 * @author wulele
 */
// 这个类被spring接管，方法返回值为String、有具体视图跳转就会被解析
@Controller
@RequestMapping("/hello")
public class HelloSpringmvc {
    /**
     * localhost:80/hello/spring
     */
    @RequestMapping("/spring")
    public String Hello(Model model){
        model.addAttribute("msg", "Hello SpringMVC");
        // /WEB-INF/jsp/hello.jsp
        return "hello";
    }
}

```

3. 运行结果同上

接收请求参数

浏览器地址栏参数名与Controller中函数参数名相同

当使用@RequestParam("参数名")时，浏览器地址栏参数名必须与注解中设置的参数名一样

Controller中函数参数为一个对象（实体类），地址栏传参必须与实体类的字段名一致

字符过滤（乱码问题）

1. 自定义过滤器，自写继承自Filter的类，注册filter，详见[smbms学习笔记（服务器端） - 芜湖男酮 - 博客园 \(cnblogs.com\)](#)
2. 使用SpringMVC提供的字符过滤，配置在web.xml

```
<filter>
  <filter-name>encoding</filter-name>
  <filter-
class>org.springframework.web.filter.CharacterEncodingFilter<
/filter-class>
  <init-param>
    <param-name>encoding</param-name>
    <param-value>utf-8</param-value>
  </init-param>
</filter>
<filter-mapping>
  <filter-name>encoding</filter-name>
  <!-- 包括.jsp -->
  <url-pattern>/*</url-pattern>
</filter-mapping>
```

3. 终极方案：使用网上大神写好的自定义过滤器

Jackson使用及JsonUtils（Json工具类）的书写

1. pom.xml

```
<dependency>
  <groupId>com.fasterxml.jackson.core</groupId>
  <artifactId>jackson-databind</artifactId>
  <version>2.12.3</version>
</dependency>
```

2. springmvc-servlet.xml

```
<!-- 注解驱动开启 json乱码处理 -->
<mvc:annotation-driven>
  <mvc:message-converters register-defaults="true">
    <bean
class="org.springframework.http.converter.StringHttpMessageCo
nverter">
      <constructor-arg value="UTF-8"/>
    </bean>
```



```

        <bean
class="org.springframework.http.converter.json.MappingJackson
2HttpMessageConverter">
        <property name="objectMapper">
            <bean
class="org.springframework.http.converter.json.Jackson2Object
MapperFactoryBean">
                <property name="failOnEmptyBeans"
value="false"/>
            </bean>
        </property>
    </bean>
</mvc:message-converters>
</mvc:annotation-driven>

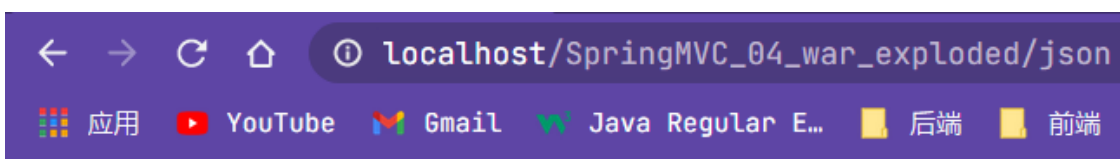
```

3. 给前端返回一个对象

```

/**
 * @author wulele
 * @RestController 不走视图解析器，直接返回字符串，作用于类
 * @ResponseBody 不走视图解析器，直接返回字符串，作用于方法
 */
@Controller
public class Hello {
    @RequestMapping("/json")
    @ResponseBody
    public String testJson(){
        String value = null;
        User user = new User("杰克", 12, "123123");
        ObjectMapper mapper = new ObjectMapper();
        try {
            value = mapper.writeValueAsString(user);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
        return value;
    }
}

```



```

{"name":"杰克","age":12,"password":"123123"}

```

4. 给前端返回一个对象列表

```

@RequestMapping("/json1")

```

```

@ResponseBody
public String testJson1() throws JsonProcessingException {
    List<User> userList = new ArrayList<>();
    User user1 = new User("杰克", 12, "123123");
    User user2 = new User("汤姆", 12, "123123");
    User user3 = new User("皮克", 12, "123123");
    userList.add(user1);
    userList.add(user2);
    userList.add(user3);
    ObjectMapper mapper = new ObjectMapper();
    String s = mapper.writeValueAsString(userList);
    return s;
}

```



[{"name": "杰克", "age": 12, "password": "123123"}, {"name": "汤姆", "age": 12, "password": "123123"}, {"name": "皮克", "age": 12, "password": "123123"}]

5. 给前端返回时间

```

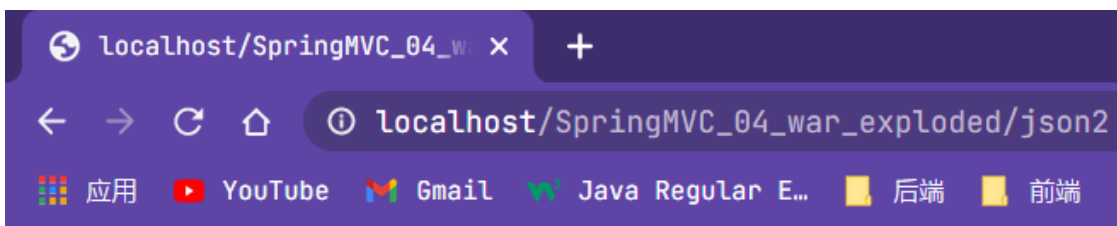
@RequestMapping("/json2")
@ResponseBody
public String testJson2() throws JsonProcessingException {
    Date date = new Date();
    ObjectMapper mapper = new ObjectMapper();
    // 关闭ObjectMapper默认日期格式为时间戳方式

    mapper.configure(SerializationFeature.WRITE_DATES_AS_TIMESTAMP, false);
    mapper.setDateFormat(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"));
    String s = mapper.writeValueAsString(date);

    // 将时间戳格式化为指定格式
    // SimpleDateFormat dateFormat = new
    SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
    // String s = dateFormat.format(date);

    return s;
}

```



"2021-05-28 11:17:29"

6. 封装为Json工具类

```

/**
 * @author wulele
 */
public class JsonUtil {

    public static String getJson(Object object){
        return getJson(object,new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"));
    }

    public static String getJson(Object object,
SimpleDateFormat simpleDateFormat){
        ObjectMapper mapper = new ObjectMapper();

        mapper.configure(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS,false);
        mapper.setDateFormat(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss"));
        try {
            return mapper.writeValueAsString(object);
        } catch (JsonProcessingException e) {
            e.printStackTrace();
        }
        return null;
    }
}

```

7. 举例优化返回给前端一个对象、时间

```

// 返回一个对象
@RequestMapping("/json")
@ResponseBody
public String testJson(){
    String value = null;
    User user = new User("杰克", 12, "123123");
    //ObjectMapper mapper = new ObjectMapper();
    //try {
    //    value = mapper.writeValueAsString(user);
    //} catch (JsonProcessingException e) {
    //    e.printStackTrace();
    //}
    return JsonUtil.getJson(user);
}

// 返回时间
@RequestMapping("/json3")
@ResponseBody
public String testJson3() throws JsonProcessingException {
    return JsonUtil.getJson(new Date());
}

```

```
}
```

FastJson使用

```
@RequestMapping("/json4")
@ResponseBody
public String testJson4(){
    // 返回对象列表
    //List<User> userList = new ArrayList<>();
    //User user1 = new User("杰克", 12, "123123");
    //User user2 = new User("汤姆", 12, "123123");
    //User user3 = new User("皮克", 12, "123123");
    //userList.add(user1);
    //userList.add(user2);
    //userList.add(user3);
    //String s = JSON.toJSONString(userList);
    // 返回时间
    String s = JSON.toJSONString(new
Date(),SerializerFeature.DisableCircularReferenceDetect,
SerializerFeature.WriteDateUseDateFormat);
    return s;
}
```

HandlerInterceptor

实现HandlerInterceptor接口的方法（不是必须实现），类比Filter[smbms学习笔记（服务器端）](#) - 芜湖男酮 - 博客园 (cnblogs.com)

```
/**
 * @author wulele
 */
public class MyInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(HttpServletRequest request,
        HttpServletResponse response, Object handler) throws Exception {
        System.out.println("before");
        //true请求继续执行, false则相反
        return true;
    }

    @Override
    public void postHandle(HttpServletRequest request,
        HttpServletResponse response, Object handler, ModelAndView
        modelAndView) throws Exception {
```

```
        System.out.println("after");
    }

    @Override
    public void afterCompletion(HttpServletRequest request,
        HttpServletResponse response, Object handler, Exception ex) throws
        Exception {
        System.out.println("clear");
    }
}
```

springmvc-servlet.xml

```
<mvc:interceptors>
    <mvc:interceptor>
        <!-- 过滤所有请求 -->
        <mvc:mapping path="/**" />
        <bean class="com.wll.filter.LoginInterceptor" />
    </mvc:interceptor>
</mvc:interceptors>
```