**SpringMVC**

# 组件

## 底层组件调度顺序

### DispatcherServlet获得请求后会分别调用下面的组件

## HandlerMapping

### 处理器映射器

* 返回处理器执行链HandlerExecutionChain

## HandlerAdaptor

### 处理器适配器

* 返回ModelAndView

## ViewResolver

### 视图解析器

* 返回对应的view视图对象

## 最后渲染视图返回客户端

# 配置文件

## context:component-scan标签

### 扫描@controller注解

## Bean

### 内部资源视图解析器

<bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver"> <property name="prefix" value="/user/"/> <property name="suffix" value=".jsp"/></bean>

* prefix

• 前缀

* suffix

• 后缀

* 为解释器手动添加固定的前后缀

# 使用

## 1.导入相关依赖

<dependency> <groupId>org.springframework</groupId> <artifactId>spring-webmvc</artifactId> <version>5.3.10</version></dependency>

## 2.配置前端控制器Servlet

<init-param> <param-name>contextConfigLocation</param-name> <param-value>classpath:spring-mvc.xml</param-value></init-param>

### 需要导入一个spring-mvc.xml的配置文件

* 里面需要配置扫描Controller的包

## 3.写Controller类，需要加上@Controller注解

### 给方法加上@RequestMapping注释映射

* value为访问的路径

• 方法返回一个string类型，他是对应的资源名称

• 例如index.jsp或者spring\_web这个servlet

• forward:放在字符串前面表示转发

• redirect:放在字符串前面表示重定向

## 4.创建spring.mvc.xml

### 用于配置扫描Controller类的标签

## 5.记得配置前端控制器的servlet的时候使用init-param标签传入spring-mvc.xml的位置

## 6.最后访问@ReqeustMapping映射的路径即可访问到返回的资源

# 获取请求数据

## 1.当方法的形参名和传入的key相同，就会为它赋值

### 如果没有传入对应的参数会爆错

## 2.形参是标准Bean类，那么会调用传入的key对应的set为对象内部赋值

### 如果是Bean，那么名称对应的会赋值，不对应的或者空参数就不会赋值也不会爆错

## 3.形参是一个数组，那么会接收和数组名称一样的参数vlaue

## 4.需要存储集合

### 需要用到一个对象将集合包裹起来，然后向对象里面赋值

* <input type="text" name="userList[0].name">

• 表示向这个对象里的userList集合的第0个对象的name赋值

### 如果直接使用集合作为形参需要在形参前面加个@RequestBody注解

@RequestBody表示将请求体直接封装到形参中

* 那么前端需要ajax发送application/JSON格式的数据

• 数据里面为一个Array对象，并且按照后端形参中的集合的存放类属性编写

# MVC静态资源的处理

## 当前端需要获取静态资源的时候，如果没有配置Servlet那么会按照MVC的前端控制器分配，回去找对应的@RequestMapping，所以一般会爆错

## 在Spring\_mvc.xml中配置

### 开放对应的资源访问

* 一旦虚拟路径为这个，那么就默认为资源

### <mvc:resources mapping="/img/\*\*" location="/img/"/>

* 映射虚拟路径为location对应的路径

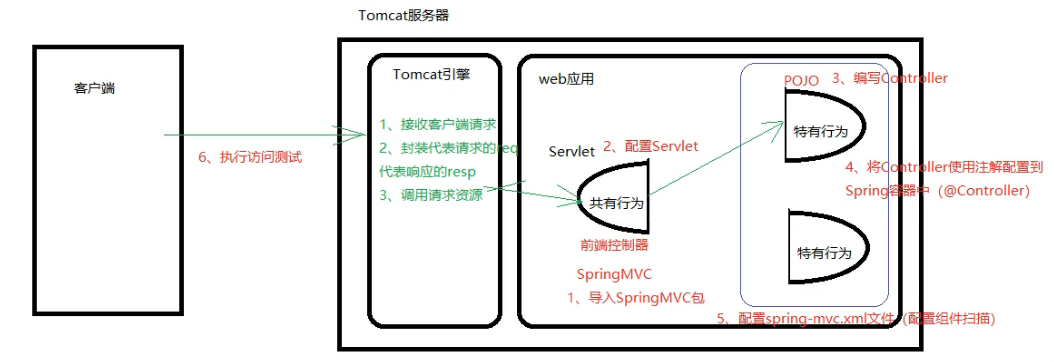
### <mvc:default-servlet-handler/>

* 当前端控制器找不到controller，交由原始中间件处理请求

# 概念

## 将Web层的共有代码提取出来，由一个前端控制器来调用指定行为

### 



# 注解

## RequestMapping

### value

* 表示访问路径

### method

* 表示访问的请求方式

• 是一个枚举类型

### params

* 限制请求参数，必须携带的请求参数

• params = {"name","password!123"

• 表示必须携带name和password，并且pw不能为123

# 数据响应

## 页面跳转

### 返回一个字符串

* 字符串代表资源路径

### 返回一个ModelAndView对象

@RequestMapping("/quick6")public ModelAndView save6() { ModelAndView modelAndView = new ModelAndView("success.jsp"); modelAndView.addObject("name", "2657944563"); return modelAndView;}

* 这个对象可以添加数据和设置视图

• 添加的数据可以在对应的视图使用EL表达式获取

• 视图就是对应的展示页面

### MVC注入

* 当@RequestMapping的方法需要一个参数的时候MVC会注入这个参数然后提供调用

• Model

• 提供数据写入源

• 提供给页面源提取

• ModelAndView

• 提供完整的ModelAndView

• 但是里面是空的需要手动setViewName和Attribute

* 大部分在编写Servlet的时候需要的对象MVC都准备好了，只需要在方法形参写就行了，MVC会自动注入

• 但是为了解耦不常用

## 回写数据

### 使用MVC注入的Resopnse对象进行回写

### 使用@ResponseBody

@RequestMapping("/quick10")@ResponseBodypublic String save10() { return "@Response注解回写数据测试";}

* 告知MVC框架这个方法作为回写数据

### 当需要返回对象或集合的时候需要配置请求映射适配器

<bean class="org.springframework.web.servlet.mvc.method.annotation.RequestMappingHandlerAdapter"> <property name="messageConverters"> <list> <bean class="org.springframework.http.converter.json.MappingJackson2HttpMessageConverter"/> </list> </property></bean>

* 配置完成后返回一个标准Bean对象会回写Json格式的数据
* xml注解配置

• <mvc:annotation-driven/>

• 需要引入命名空间MVC