链滴

Spring 构建 Web 应用

作者：jiangyue

原文链接：https://ld246.com/article/1479176053986

来源网站：链滴

Spring MVC 中，请求会首先通过一个单实例的前端控制器--DispatcherServlet，它的任务是查询处理器映射(handler mapping)，并将请求发给 Spring MVC 控制器(controller)。

经过逻辑处理产生的信息被称作模型(model)。控制器获取用户提交的信，将逻辑处理后得到的模型数据打包，并标示出用于渲染输入的视图，发送到 DispatcherServlet。良好的控制器本身只处理很少甚至不理工作，而将业务逻辑委托给一个或多个服务对象。

DispatcherServlet 使用视图解析器(view resolver) 将视图名匹配为特定视图实现，交付模型数据。视图使用模型数据渲染输出，通过相应对象传递给客户端。

借助 Servlet3 规范和 Spring3.1 的功能增强，可以使用 Java 将 DispatcherServlet 配置在 Servlet 容器中，而不使用 web.xml。

扩展 <code>AbstractAnnotationConfigDispatcherServletInitializer</code> 的任意类会自配置 DispatcherServlet 和 Spring 应用上下文。

扩展类的 <code>getServletMappings()</code> 方法将一个或多个路径映射到 DispatcherServlet 上。<code>/</code>表示它会是应用的默认 Servlet，处理进入应用的有请求。

DispatcherServlet 启动时会创建 Spring 应用上下文。扩展类的 <code>getServletConfigClasses()</code> 方法定义 DispatcherServlet 加载的应用上下文时，加载在方法中添加的配置类的 bean 包含 web 组件的 bean，如控制器、视图解析器等)。

DispatcherServlet 加载包含 web 组件的 bean，而 ContextLoaderListener 要加载应用中的其他 bean。如驱动应用后端的中间层和数据层组件。

扩展类的 <code>getServletConfigClasses()</code> 方法通过返回带有 <code>@Configuration</code> 注解的类(WebConfig)来定义 DispatcherServlet 应用上下文中的 bean，扩展类的 <code>getRootConfigClasses()</code> 方法通过返回带有 <code>@Configuration</code> 注解类(RootConfig)来配置 ContextLoaderListener 创建的应用上下文中的 bean。

通过 <code>AbstractAnnotationConfigDispatcherServletInitializer</code> 来配置 DispatherServlet 是传统 web.xml 方式的替代方案，只能部署到支持 Servlet3.0 的服务器中，如 Tomcat7 或更高的版本。

在 Spring 的 xml 配置中，可以使用 <code><mvc:annotation-driven></code> 启动解驱动的 Spring MVC。

在 Java 配置中，可以使用 <code>@EnableWebMvc</code> 注解修饰类来启动 Spring MV
。

WebConfig 扩展 <code>WebMvcConfigurerAdapter</code> 类并重写 <code>configure
efaultServletHandling()</code> 方法，调用 <code>DefaultServletHandlerConfigurer</code>
的 <code>enable()</code> 方法，要求 DispatcherServlet 将静态资源的请求转发到 Se
vlet 容器的默认 Servlet 上，而不使用 DispatcherServlet 本身处理此类请求。

在 Spring MVC 中，控制器只是在方法上添加 <code>@RequestMapping</code> 注解的类
注解声明了所要处理的请求。方法返回 String 类型的视图名称。

<code>@Controller</code> 注解基于 <code>@Component</code> 注解，目的是实现
件扫描。

将路径转移到类级别的 <code>@RequestMapping</code> 上可以定义类级别的请求处理，
法上的 <code>@RequestMapping</code> 注解会对类级别的注解进行补充。

<code>@RequestMapping</code> 注解的 value 属性可以接受一个 <s
rong>String 数组，用来映射到多个路径的请求。

在 <code>@RequestMapping</code> 注解修饰的方法中，可以添加 Model</str
ng> 类型的参数。调用 Model 的 <code>addAttribute ()</code>方法，将数据存入 Model 中。
odel 实际上是一个 Map，会传递给视图，从而渲染到客户端。

如未指定 <code>addAttribute()</code> 的 key，模型 key 会根据<str
ng>数据类型推断出。如未返回视图名(直接返回了数据)，逻辑视图的名称</str
ng>会根据请求路径推断。

Spring MVC 允许多种方式将客户端中的数据传递到控制器中处理，包括：查询参数(
uery Parameter)，表单参数(Form Parameter)和路径变
(Path Variable)。

<code>@RequestParam</code> 注解用来修饰方法中的参数，用来获
请求参数。利用注解的 defaultValue 属性，可以在未获取到请求参数时<stron
>指定默认值。因请求参数都是 String 类型，所以指定的默认值需
转换为 String 类型。

在理想情况下，要识别资源应该通过 URL 路径进行标示，而不是查询参数
Spring MVC 允许在 <code>@RequestMapping</code> 的路径中添加占位符</stron
>(用<code>{}</code>括起来)，在方法中使用 <code>@PathVariable</code> 获取路径变量。如
<code>@PathVariable</code> 未指定 value 值，会假设占位符
称与方法参数名相同。

<code><form></code> 表单未指定 action 时，会默认提交到展示时的 URL 路径上。

<code>@RequestMapping</code> 修饰的方法中，可以添加自定义类型的对象作为参数。对象的属性会使用请求中同名的参数进填充。

<code>InternalResourceViewResolver</code> 会识别控制器方法返回字符串中的 <code>redirect:</code> 或 <code>forward:</code> 确定要将其解析为 重定向/前往规则。

从 Spring3.0开始，Spring MVC 提供了对 Java 校验 API 的支持，而不需要特殊配置。

在获取请求参数的自定义类中，使用校验参数的注解，如 <code>@NotNull</code>、<code>@size(min=5, max=25)</code>等修饰属性，并在控制器方法参数前使用 <code>@Valid</code> 注解开启校验功能。在控制器方法中添加 Errors 类型的参数，即可在方法中使它的 <code>hasErrors()</code> 方法获得校验结果。需要注意的一点，Errors 参数要紧跟在带有 <code>@Valid</code> 注解的参数后面。

