一、首先，我们先来认识一下SpringMVC的主要组件

前端控制器（DisatcherServlet）:接收请求，响应结果，返回可以是json,String等数据类型，也可以是页面（Model）。

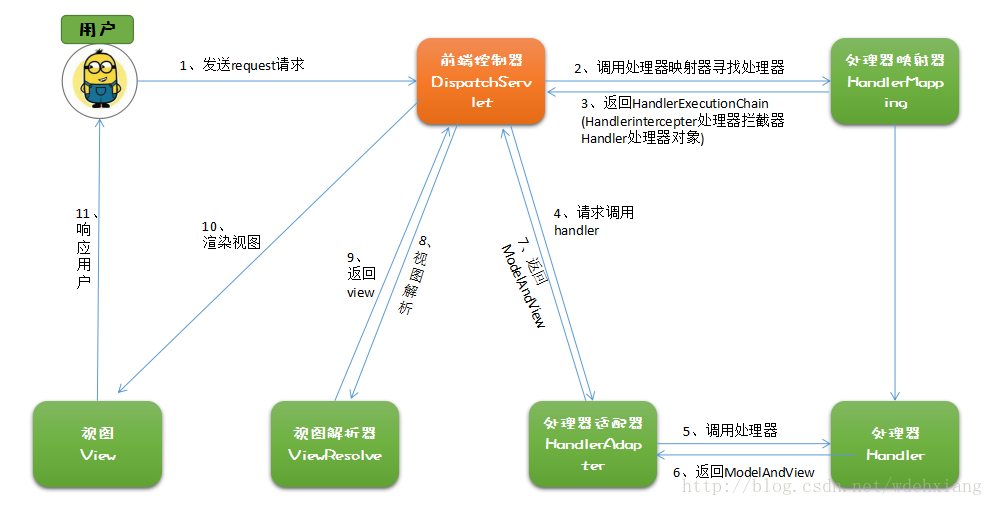
处理器映射器（HandlerMapping）:根据URL去查找处理器，一般通过xml配置或者注解进行查找。

处理器（Handler）：就是我们常说的controller控制器啦，由程序员编写。

处理器适配器（HandlerAdapter）:可以将处理器包装成适配器，这样就可以支持多种类型的处理器。

视图解析器（ViewResovler）:进行视图解析，返回view对象（常见的有JSP,FreeMark等）。

二、SpingMVC的工作原理



下面是文字步骤说明：

1、用户发送请求到前端控制器（DispatcherServlet）。

2、前端控制器请求处理器映射器（HandlerMapping）去查找处理器（Handler）。

3、找到以后处理器映射器（HandlerMappering）向前端控制器返回执行链（HandlerExecutionChain）。

4、前端控制器（DispatcherServlet）调用处理器适配器（HandlerAdapter）去执行处理器（Handler）。

5、处理器适配器去执行Handler。

6、处理器执行完给处理器适配器返回ModelAndView。

7、处理器适配器向前端控制器返回ModelAndView。

8、前端控制器请求视图解析器（ViewResolver）去进行视图解析。

9、视图解析器向前端控制器返回View。

10、前端控制器对视图进行渲染。

11、前端控制器向用户响应结果。

|  |
| --- |
| <!-- 如果是用mvn命令生成的xml，需要修改servlet版本为3.1 -->  <!-- 配置DispatcherServlet -->  <servlet>  <servlet-name>dispatcher</servlet-name>  <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  <init-param>  <param-name>contextConfigLocation</param-name>  <param-value>classpath:spring/spring-\*.xml</param-value>  </init-param>  </servlet>  <servlet-mapping>  <servlet-name>dispatcher</servlet-name>  <!-- 默认匹配所有的请求 -->  <url-pattern>/</url-pattern>  </servlet-mapping> |

|  |
| --- |
| <!-- 配置SpringMVC -->  <!-- 1.开启SpringMVC注解模式 -->  <!-- 简化配置：  (1)自动注册DefaultAnootationHandlerMapping,AnotationMethodHandlerAdapter  (2)提供一些列：数据绑定，数字和日期的format @NumberFormat, @DateTimeFormat, xml,json默认读写支持 -->  **<mvc:annotation-driven />**  <!-- 2.静态资源默认servlet配置 (1)加入对静态资源的处理：js,gif,png (2)允许使用"/"做整体映射 -->  <mvc:default-servlet-handler />  <!-- 3.配置jsp 显示ViewResolver -->  <bean  class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>  <property name=*"viewClass"*  value=*"org.springframework.web.servlet.view.JstlView"* />  <property name=*"prefix"* value=*"/WEB-INF/jsp/"* />  <property name=*"suffix"* value=*".jsp"* />  </bean>  <!-- 4.扫描web相关的bean -->  **<context:component-scan base-package=*"com.ssm.controller"* />** |

注解扫描包,一般要扫描controller

<!-- 启用spring mvc 注解 -->

<context:annotation-config />

<!-- 设置使用注解的类所在的包 -->

<context:component-scan base-package="controller"></context:component-scan>

也要扫描dao和service

<!-- 启用注解 -->

<context:annotation-config />

<!-- 设置使用注解的类所在的包 -->

<context:component-scan base-package="dao,service.\*"></context:component-scan>

<mvc:annotation-driven /> 是一种简写形式，完全可以手动配置替代这种简写形式，简写形式可以让初学都快速应用默认配置方案。

**<mvc:annotation-driven /> 会自动注册DefaultAnnotationHandlerMapping与AnnotationMethodHandlerAdapter 两个bean,是spring MVC为@Controllers分发请求所必须的。**  
并提供了：数据绑定支持，@NumberFormatannotation支持，@DateTimeFormat支持，@Valid支持，读写XML的支持（JAXB），读写JSON的支持（Jackson）。  
后面，我们处理响应ajax请求时，就使用到了对json的支持。  
后面，对action写JUnit单元测试时，要从spring IOC容器中取DefaultAnnotationHandlerMapping与AnnotationMethodHandlerAdapter 两个bean，来完成测试，取的时候要知道是<mvc:annotation-driven />这一句注册的这两个bean。

在基于主机方式配置Spring的配置文件中，你可能会见到**<context:annotation-config/>**这样一条配置，他的作用是式地向 Spring 容器注册

AutowiredAnnotationBeanPostProcessor、CommonAnnotationBeanPostProcessor、PersistenceAnnotationBeanPostProcessor 以及 RequiredAnnotationBeanPostProcessor 这 4 个BeanPostProcessor。

注册这4个BeanPostProcessor的作用，就是为了你的系统能够识别相应的注解。

如果你想使用@Autowired注解，那么就必须事先在 Spring 容器中声明 AutowiredAnnotationBeanPostProcessor Bean。传统声明方式如下

<bean class="org.springframework.beans.factory.annotation. AutowiredAnnotationBeanPostProcessor "/>

如果想使用@ Resource 、@ PostConstruct、@ PreDestroy等注解就必须声明CommonAnnotationBeanPostProcessor

如果想使用@PersistenceContext注解，就必须声明PersistenceAnnotationBeanPostProcessor的Bean。

如果想使用 @Required的注解，就必须声明RequiredAnnotationBeanPostProcessor的Bean。同样，传统的声明方式如下：

<bean class="org.springframework.beans.factory.annotation.RequiredAnnotationBeanPostProcessor"/>

一般来说，这些注解我们还是比较常用，尤其是Antowired的注解，在自动注入的时候更是经常使用，所以如果总是需要按照传统的方式一条一条配置显得有些繁琐和没有必要，于是spring给我们提供<context:annotation-config/>的简化配置方式，自动帮你完成声明。

不过，呵呵，我们使用注解一般都会配置扫描包路径选项

<context:component-scan base-package=”XX.XX”/>

该配置项其实也包含了自动注入上述processor的功能，**因此当使用 <context:component-scan/> 后，就可以将 <context:annotation-config/> 移除了。**

最后的配置如果没有<mvc:annotation-driven/>，那么所有的Controller可能就没有解析，所有当有请求时候都没有匹配的处理请求类，就都去<mvc:default-servlet-handler/>即default servlet处理了。

添加上<mvc:annotation-driven/>后，相应的do请求被Controller处理，而静态资源因为没有相应的Controller就会被default servlet处理。总之没有相应的Controller就会被default servlet处理就ok了。

后来为了解决静态资源访问的问题，servlet改成了拦截所有请求，即/，并添加了默认的servlet，这时候\*.do请求不能被控制器捕捉了，页面错误为404。直到添加了<mvc:annotation-driven/>之后，.do请求才又能被正确捕捉和处理。

可以没有

**<context:annotation-config/>**

但必须有

**<context:component-scan/>**

**<mvc:annotation-driven />**