## 第一章 SpringMVC概述

### 什么是SpringMVC

它是基于MVC开发模式的框架，用来优化控制器，它是Spring家族的一员，他也具备IoC和AOP

#### 什么是MVC？

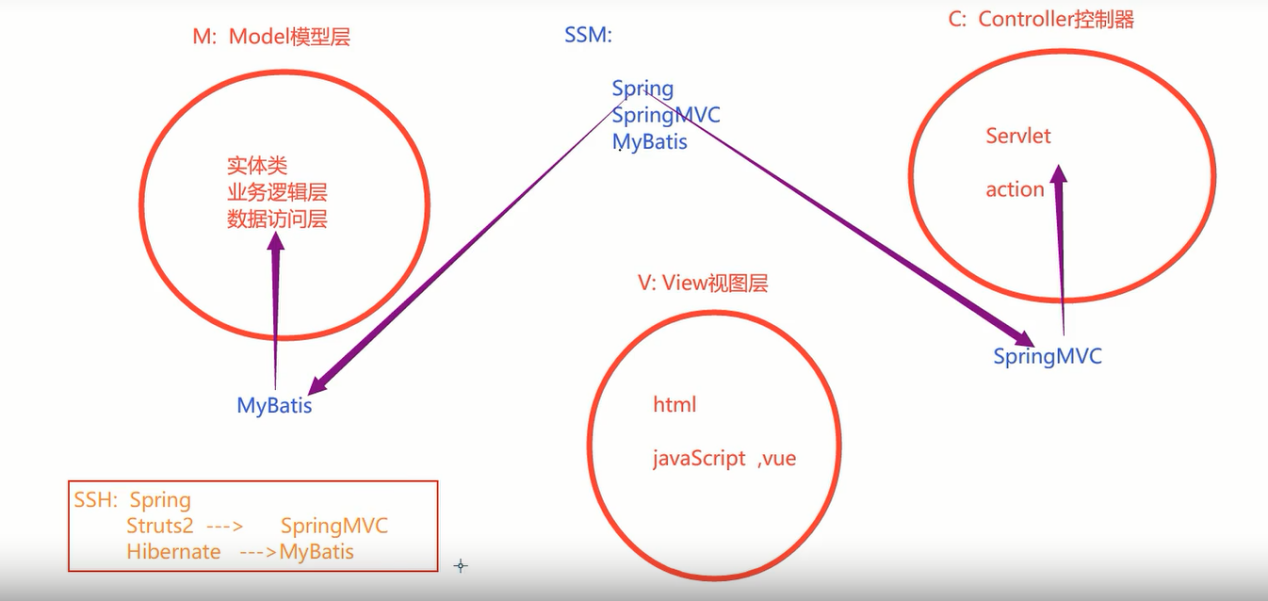
他是一种开发模式，它是模型视图控制器的简称，所以的web应用都是基于MVC开发的：

M:模型层，包含实体类，业务逻辑层，数据访问层

V:视图层, html，javascript , vue等都是视图层,用来显现数据

C:控制器，它是用来接收客户端的请求,并返回响应到客户端的组件,servlet就是组件

MyBatis简化了数据访问层，SpringMVC简化了Servlet，Spring整合了MyBatis和SpringMVC框架，使开发更加简单



#### SpringMVC的优点

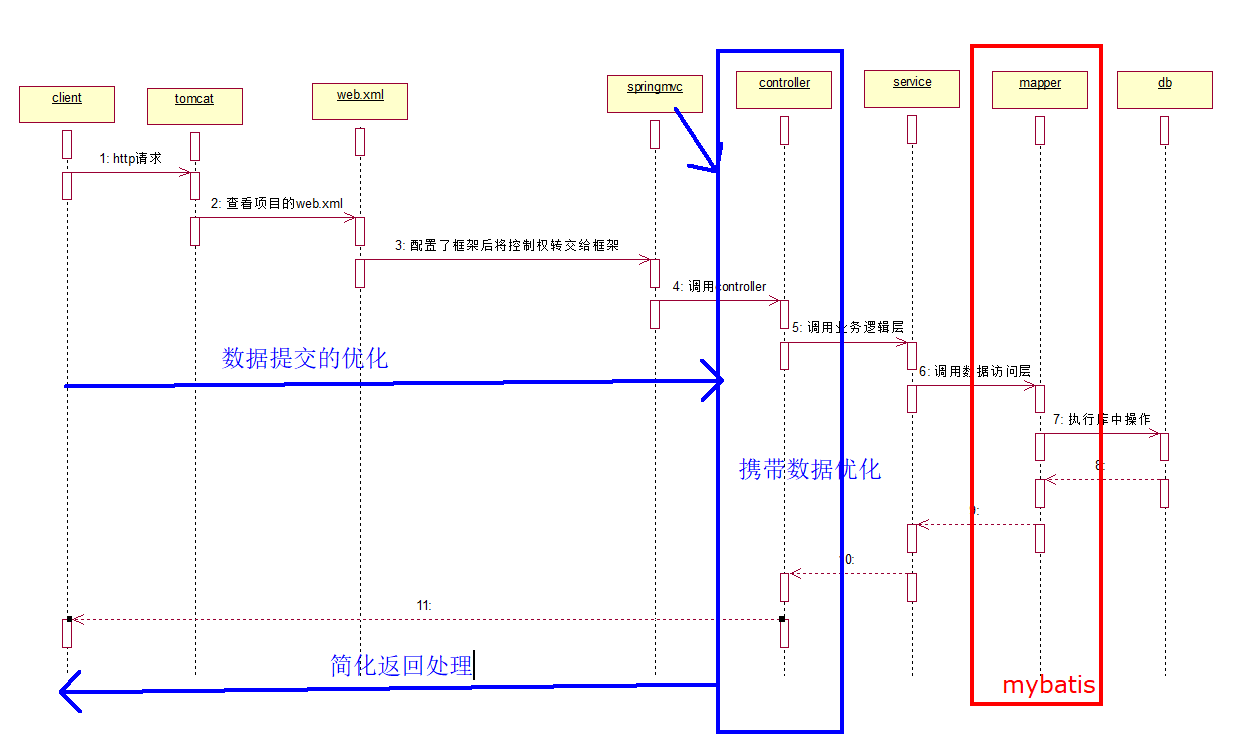
轻量级，基于MVC的框架

易于上手，容易理解，功能强大

具备IoC和AOP

完全基于注解开发

如下图，SpringMVC控制的就是controller，蓝框这一部分：



当客户端发送请求，tomcat（服务器）拦截到请求，会去查看web.xml文件，发现如果我们配置了框架，就会将控制权交给MVC框架

### SpringMVC框架的开发步骤

l)新建maven项目，选择webapp模板．

2)修改目录，添加缺失的test,java ,resources(两套)，并修改目录属性

3)修改pom. xml文件,添加springMVC的依赖，添加servlet的依赖

4)添加springmvc.xml配置文件，指定包扫描,添加视图解析器.

5)删除web.xml文件（因为版本太老），新建web.xml

6)在web.xml文件中注册springMVC框架(所有的web请求都是基于servlet的，而springMVC框架中的请求只是一个类，后面会讲怎么操作)

7)删除index.jsp页面，并新建，发送请求给服务器

8)开发控制器(Servlet)，它是一个普通的类．

9)添加tomcat进行测试功能

### 第一个demo程序

新建空项目，加入模组，选web-app模板

这个模板需要我们手动添加test文件夹，以及com.bjpowernode包，还有resources包

#### pom.xml文件的设置

目录补齐之后修改pom.xml文件

添加依赖：

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-webmvc</artifactId>

<version>5.2.5.RELEASE</version>

</dependency>

<dependency>

<groupId>javax.servlet</groupId>

<artifactId>javax.servlet-api</artifactId>

<version>3.1.0</version>

<scope>provided</scope>

</dependency>

删除多余配置，然后添加资源文件位置：

<resources>

<resource>

<directory>src/main/java</directory>

<includes>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.properties</include>

</includes>

</resource>

<resource>

<directory>src/main/resources</directory>

<includes>

<include>\*\*/\*.xml</include>

<include>\*\*/\*.properties</include>

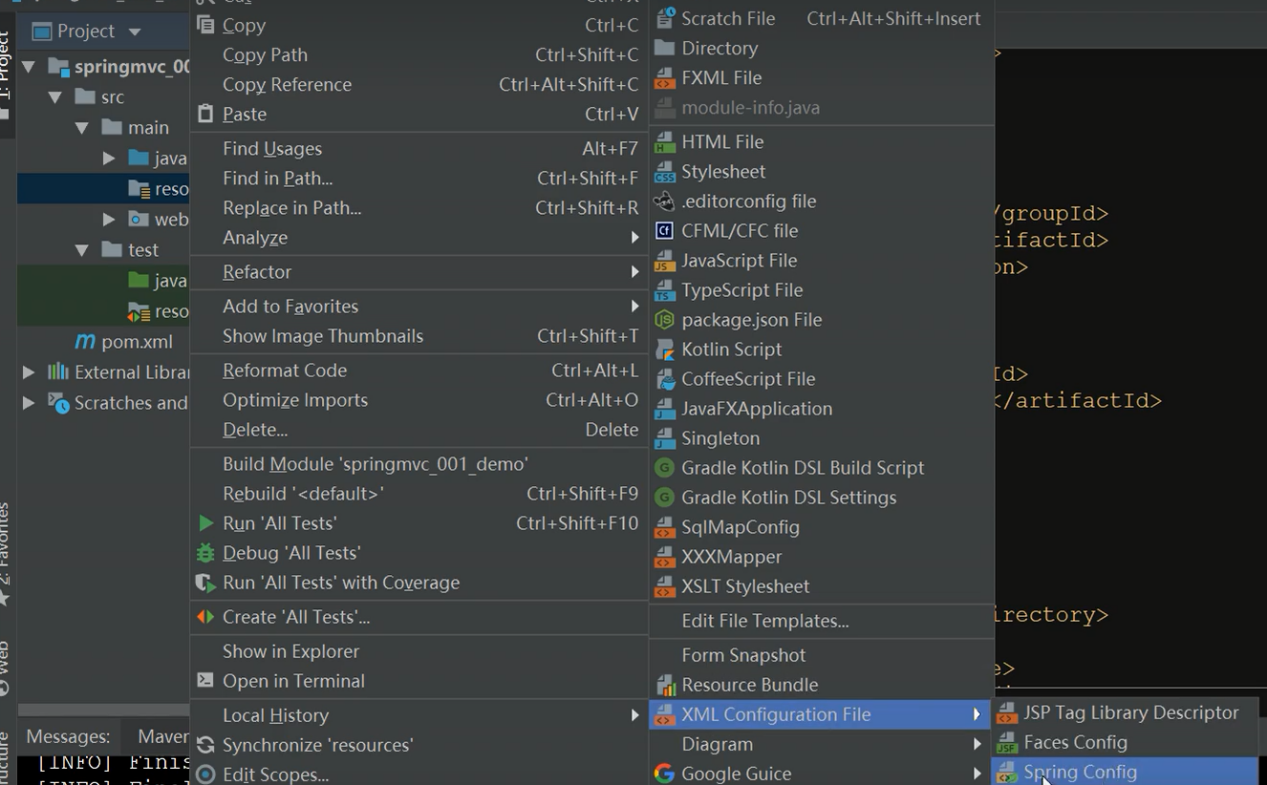
</includes>

</resource>

</resources>

#### springmvc.xml文件的设置

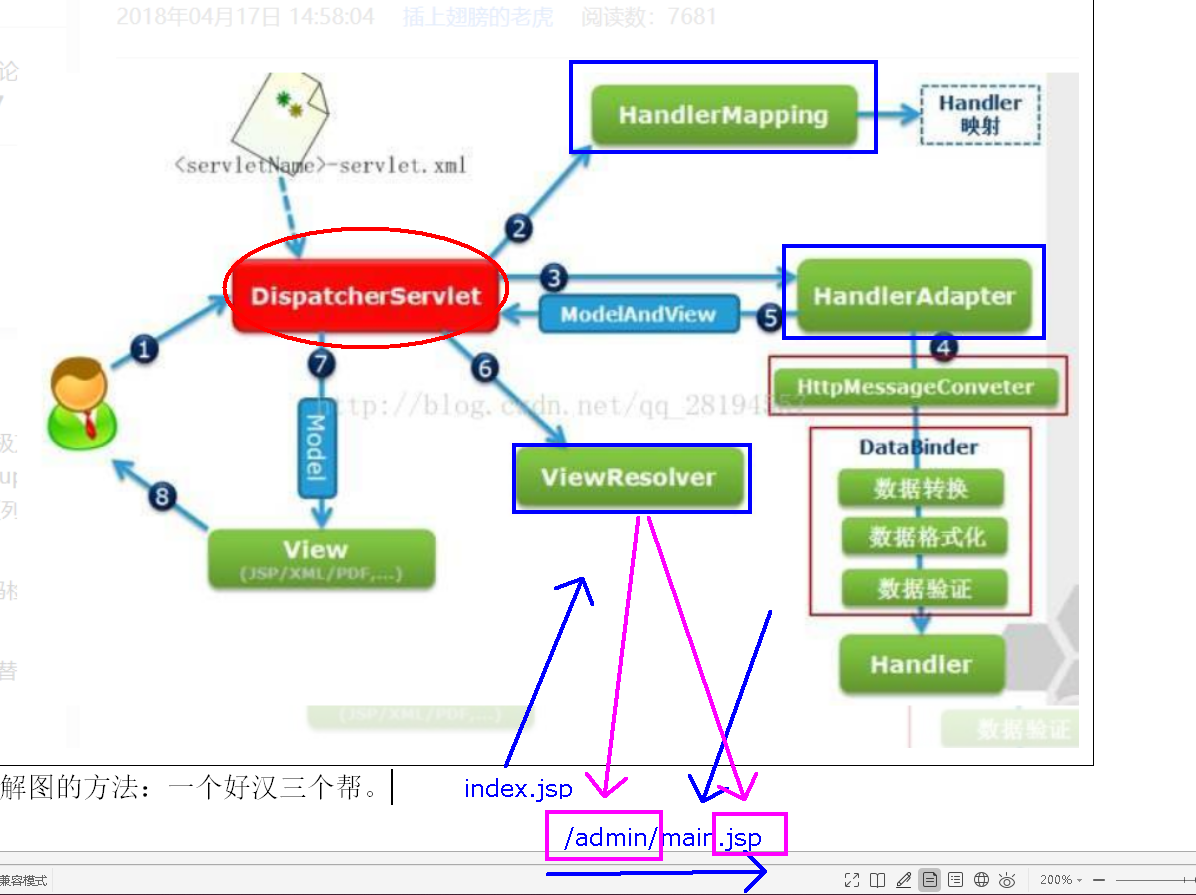
然后添加springmvc.xml文件，名字其实可以随意起，但是我们规范一些，就叫springmvc.xml



里面要添加包扫描

添加视图解析器

视图解析器就是下图中的ViewResolver，视图解析器要提供前缀和后缀，就是图中的 /admin/（前缀） 和 .jsp （后缀）,为此我们在webapp文件夹下创建admin文件夹，其中再创建main.jsp文件



目前xml文件中我们写的内容是：

<context:component-scan base-package="com.bjpowernode.controller">

</context:component-scan>  
<!-- 添加视图解析器 -->  
<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">  
<!-- 配置前后缀 -->  
 <property name="prefix" value="/admin/"></property>  
 <property name="suffix" value=".jsp"></property>  
</bean>

加了前缀后缀，我们的方法以后只需要提供main，它就是知道是/admin/main.jsp

#### web.xml文件的配置

下面我们要在web.xml文件中注册springMVC框架

先分析一下web请求：

web请求的执行流程：

index.jsp<------------------>servlet

当jsp页面发出一个请求，应该是由servlet去处理

现在使用SpringMVC则是

index.jsp<------------------>SpringMVC（SpringMVC的处理器是一个普通的方法，没有servlet怎么办？）

实际上使用SpringMVC的情况是这样的：

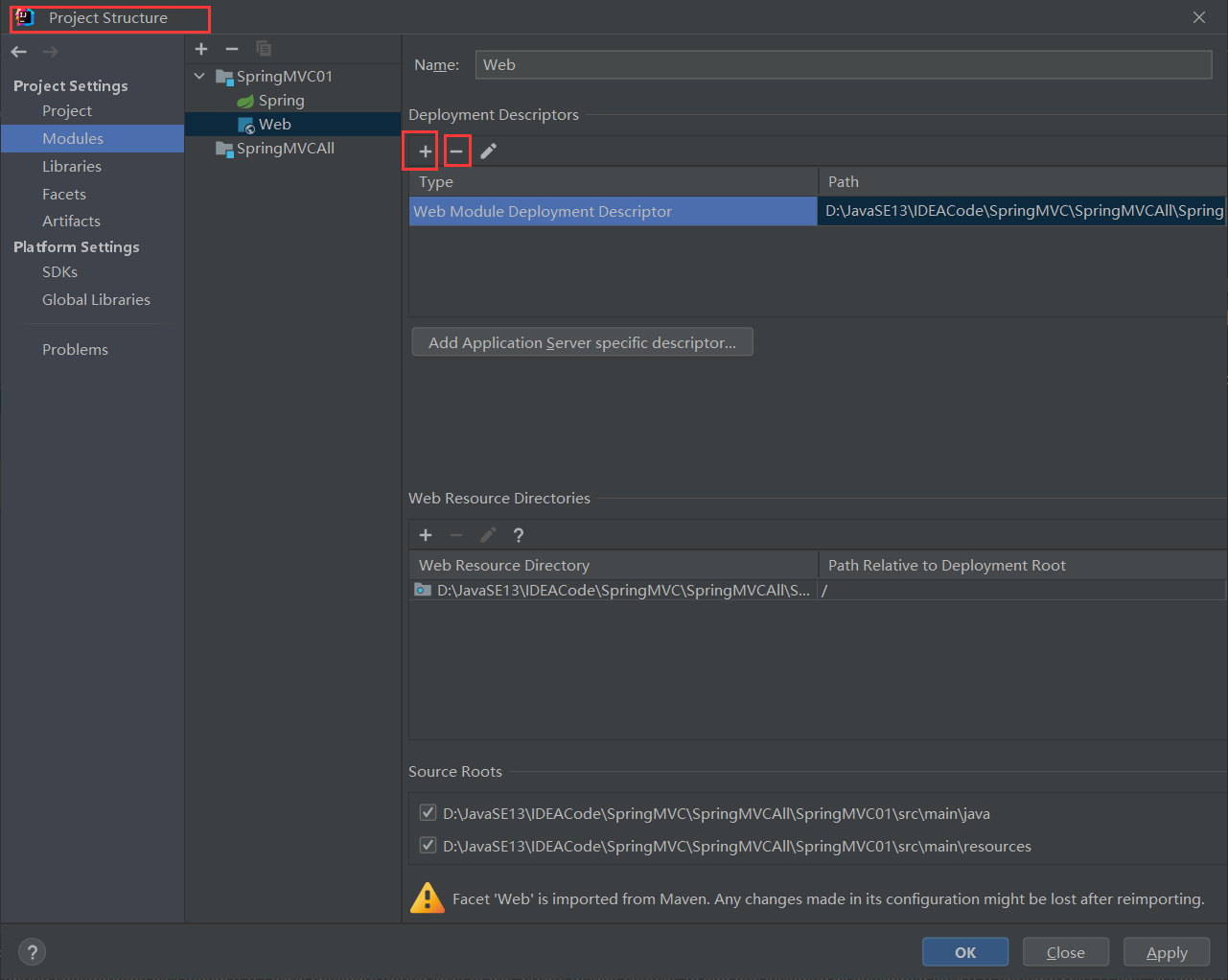
index.jsp<------------------>DispatcherServlet<----------------->SpringMVC的处理器(其实是一个普通的方法)

仅有一个普通的类中的方法是肯定搞不定处理请求这一工作的，所以还是需要一个servlet，即DispatcherServlet，由DispatcherServlet将请求转发到这个普通方法上，处理完之后方法将结果返回DispatcherServlet，再由DispatcherServlet发给网页

所以现在依旧满足所有web请求都交给servlet处理的原则

DispatcherServlet要在web.xml文件中注册才可以使用

我们先把原来的WEB\_INF下的web.xml文件删掉，打开project structure，在里面删，然后新建一个，如果创建不出来，就取个别的名字先，如web1.xml,创建好之后再改名为web.xml



搞好之后我们在里面注册SpringMVC框架

注册框架其实就是配置servlet

<!-- 注册SpringMVC框架 -->  
 <servlet>  
 <servlet-name>springmvc</servlet-name> <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 </servlet>  
 <servlet-mapping>  
 <servlet-name>springmvc</servlet-name>  
 <url-pattern>\*.action</url-pattern>  
 </servlet-mapping>

解释一下:

servlet与servlet-mapping中的servlet-name标签的内容要一样，这样它们才能关联起来。servlet-mapping标签配置的是拦截哪些请求，servlet标签用来配置哪个servlet来给我干活

因为一个xml文件中可能有多套servlet和servlet-mapping，就是通过servlet-name来绑定

url-pattern设置的是拦截什么样的请求，我们写的是\*.action

客户端或网页提交请求的方式有http://localhost:8080/one

或者http://localhost:8080/index.jsp

但是我们使用的DispatcherServlet不拦截这些请求，它拦截的请求后面都会带一个.action,也就是说http://localhost:8080/demo.action这种请求才会被拦截

当我们访问服务器的时候，这么写就不行：

<a href="${pageContext.request.contextPath}/demo">访问服务器</a>

要写成

<a href="${pageContext.request.contextPath}/demo.action">访问服务器</a>

此外，配置没有结束，更多的mvc配置我们是写在springmvc.xml文件中的，所以要告诉DispatcherServlet:

<servlet>  
 <servlet-name>springmvc</servlet-name>  
<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>  
 <init-param>  
 <param-name>contextConfigLocation</param-name>  
 <param-value>classpath:springmvc.xml</param-value>  
 </init-param>  
</servlet>

contextCongigLocation设置配置文件的位置

classpath:中填配置文件的文件名

最后删除index.jsp，再新建一个，其实就是因为老的jsp文件页面太难看，所以新建一个，页面里面就只写了一个超链接

<body>  
<br><br><br>  
<a href="${pageContext.request.contextPath}/demo.action">访问服务器</a>  
</body>

#### 新建类demoAction

建一个类似于servlet功能的处理器：DemoAction

该类也交给Spring容器去创建对象，所以加@Controller注解

action在的所有功能实现都是由缝份来完成的

action方法的规范：

1. 访问权限是public
2. 方法返回值任意