SpringMVC简单介绍及使用

1、回顾MVC三层架构

1.1、MVC三层是什么

MVC是模型(Model)、视图(View)、控制器(Controller)的简写，是一种软件设计规范。就是将业务逻辑、数据、显示分离的方法来组织代码。MVC主要作用是**降低了视图与业务逻辑间的双向偶合。**MVC不是一种设计模式，**MVC是一种架构模式。**当然不同的MVC存在差异。

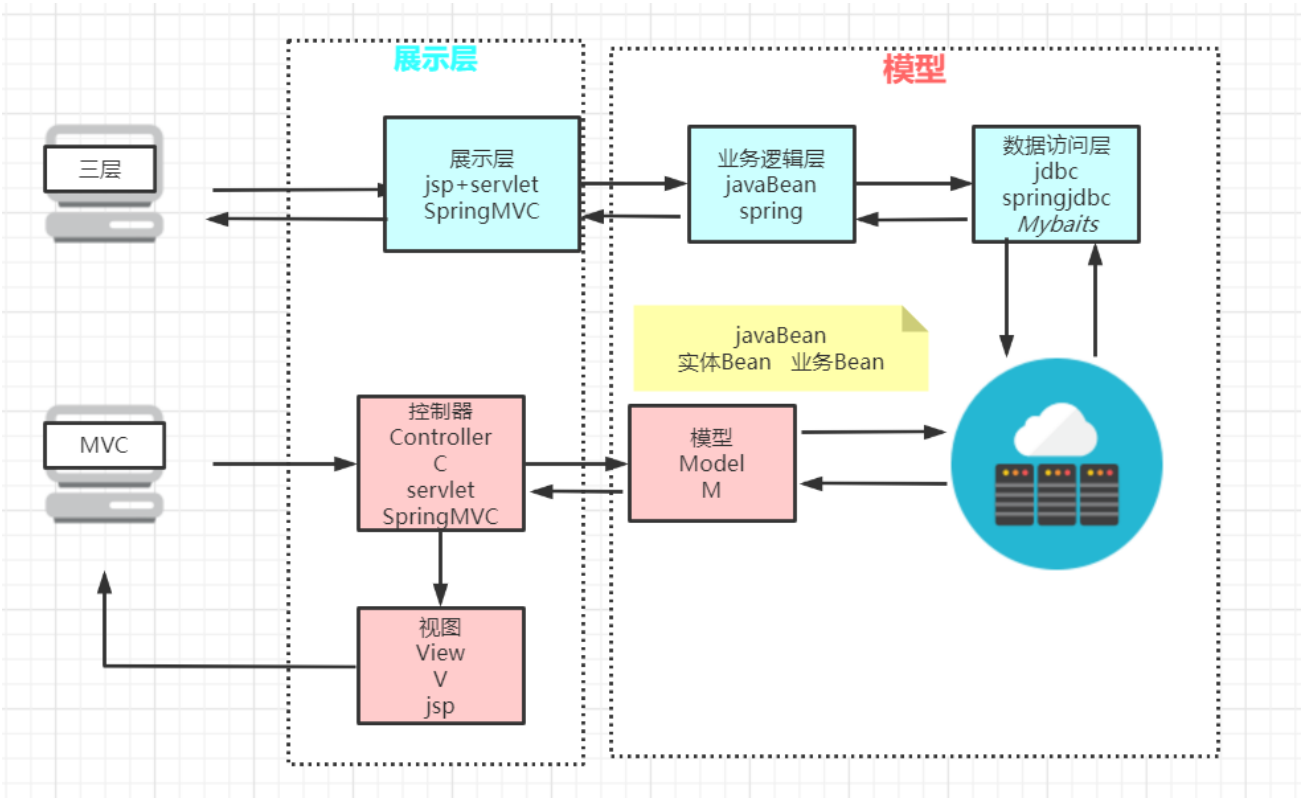
Model（模型）：数据模型，提供要展示的数据，因此包含数据和行为，可以认为是领域模型或JavaBean组件（包含数据和行为），不过现在一般都分离开来：Value Object（数据Dao） 和 服务层（行为Service）。也就是模型提供了模型数据查询和模型数据的状态更新等功能，包括数据和业务。

View（视图）：负责进行模型的展示，一般就是我们见到的用户界面，客户想看到的东西。

Controller（控制器）：接收用户请求，委托给模型进行处理（状态改变），处理完毕后把返回的模型数据返回给视图，由视图负责展示。 也就是说控制器做了个调度员的工作。

其实在最早期的时候还有model1和model2的设计模型

最典型的MVC就是JSP + servlet + javabean的模式。



servlet作为控制器的代码展示：

HelloServlet.java

```java
package cn.tulingxueyuan.controller;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

public class HelloServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletExcep
        String method = request.getParameter("method");
        if (method.equals("add")){
            request.getSession().setAttribute("msg","add");
        }else if(method.equals("sub")){
            request.getSession().setAttribute("msg","sub");
        }
        request.getRequestDispatcher("index.jsp").forward(request,response);
    }

    protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletExcept
        this.doPost(request, response);
```

```
  }
}
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
         version="4.0">

    <servlet>
        <servlet-name>HelloServlet</servlet-name>
        <servlet-class>cn.tulingxueyuan.controller.HelloServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>HelloServlet</servlet-name>
        <url-pattern>/user</url-pattern>
    </servlet-mapping>
</web-app>
index.jsp
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
 <head>
   <title>$Title$</title>
 </head>
 <body>
${msg}
 </body>
</html>
```

输入网址：http://localhost:8080/servlet_demo_war_exploded/user?method=add

2、SpringMVC

2.1、SpringMVC的介绍

> Spring Web MVC is the original web framework built on the Servlet API and has been included in the Spring Framework from the very beginning. The formal name, “Spring Web MVC,” comes from the name of its source module (spring-webmvc), but it is more commonly known as “Spring MVC”.
>
> Spring Web MVC是构建在Servlet API上的原始Web框架，从一开始就包含在Spring Framework中。 正式名称 “Spring Web MVC,” 来自其源模块(spring-webmvc)的名称，但它通常被称为“Spring MVC”。

简而言之，springMVC是Spring框架的一部分，是基于java实现的一个轻量级web框架。

学习SpringMVC框架最核心的就是DispatcherServlet的设计，掌握好DispatcherServlet是掌握SpringMVC的核心关键。

2.2、SpringMVC的优点

- 1.轻量级、可适配、非侵入，简单易学
- 2.高效，基于请求响应的MVC框架（解耦、可重用、提高维护性）
- 3.与Spring兼容性好，无缝结合

- 4.约定优于配置
- 5.功能强大：灵活的URL映射、RESTful、数据验证、格式化、本地化、主题标签库等
- 6.简洁灵活

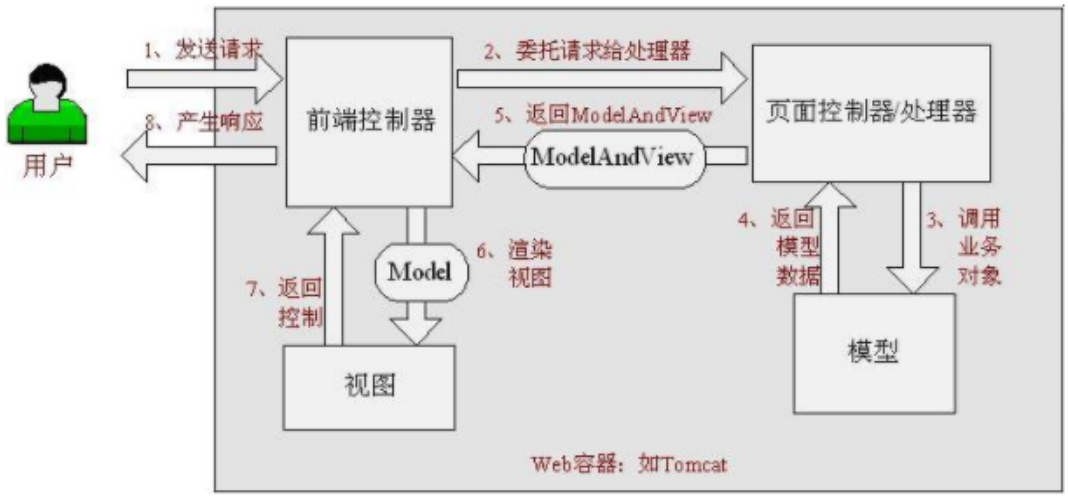
2.3、SpringMVC的实现原理

springmvc的mvc模式：

DispatcherServlet（前端控制器）

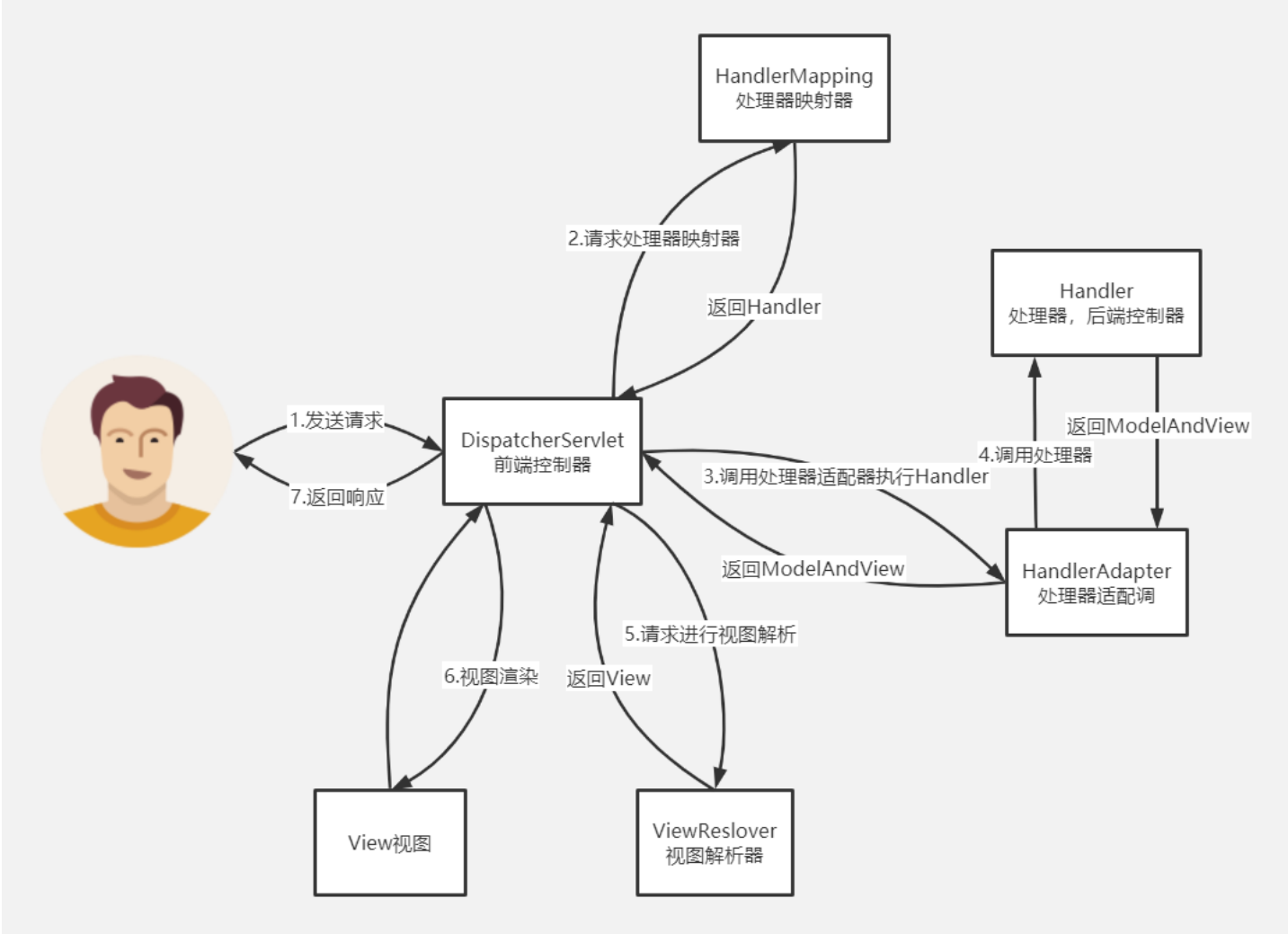
Spring的web框架围绕DispatcherServlet设计。 DispatcherServlet的作用是将请求分发到不同的处理器。从Spring 2.5开始，使用Java 5或者以上版本的用户可以采用基于注解的controller声明方式。

Spring MVC框架像许多其他MVC框架一样, 以请求为驱动 , 围绕一个中心Servlet分派请求及提供其他功能，DispatcherServlet是一个实际的Servlet (它继承自HttpServlet 基类)。



SpringMVC的具体执行流程：

当发起请求时被前置的控制器拦截到请求，根据请求参数生成代理请求，找到请求对应的实际控制器，控制器处理请求，创建数据模型，访问数据库，将模型响应给中心控制器，控制器使用模型与视图渲染视图结果，将结果返回给中心控制器，再将结果返回给请求者。



DispatcherServlet： 前端调度器， 负责将请求拦截下来分发到各控制器方法中

HandlerMapping: 负责根据请求的URL和配置@RequestMapping映射去匹配， 匹配到会返回Handler（具体控制器的方法）

HandlerAdaper: 负责调用Handler-具体的方法- 返回视图的名字 Handler将它封装到ModelAndView(封装视图名，request域的数据)

ViewReslover: 根据ModelAndView里面的视图名地址去找到具体的jsp封装在View对象中

View：进行视图渲染（将jsp转换成html内容） 最终response到的客户端

1、DispatcherServlet表示前端控制器，是整个SpringMVC的控制中心。用户发出请求，DispatcherServlet接收请求并拦截请求。

2、HandlerMapping为处理器映射。DispatcherServlet调用HandlerMapping,HandlerMapping根据请求url查找Handler。

3、返回处理器执行链，根据url查找控制器，并且将解析后的信息传递给DispatcherServlet

4、HandlerAdapter表示处理器适配器，其按照特定的规则去执行Handler。

5、执行handler找到具体的处理器

6、Controller将具体的执行信息返回给HandlerAdapter,如ModelAndView。

7、HandlerAdapter将视图逻辑名或模型传递给DispatcherServlet。

8、DispatcherServlet调用视图解析器(ViewResolver)来解析HandlerAdapter传递的逻辑视图名。

9、视图解析器将解析的逻辑视图名传给DispatcherServlet。

10、DispatcherServlet根据视图解析器解析的视图结果，调用具体的视图，进行试图渲染

11、将响应数据返回给客户端

3、基于注解的Hello_SpringMVC

1、添加pom依赖

```
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-webmvc</artifactId>
            <version>5.2.3.RELEASE</version>
        </dependency>
```

2、编写web.xml文件

```
<?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-
         version="4.0">
    <!--配置DispatcherServlet-->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <!--关联springmvc的配置文件-->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext.xml</param-value>
        </init-param>
         <load-on-startup>1</load-on-startup>
    </servlet>
    <!--匹配servlet的请求，/标识匹配所有请求-->
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <!--/*和/都是拦截所有请求，/会拦截的请求不包含*.jsp,而/*的范围更大，还会拦截*.jsp这些请求-->
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

3、编写springmvc需要的spring配置文件，applicationContext.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:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/


    <context:component-scan base-package="cn.tulingxueyuan"></context:component-scan>
</beans>
```

4、HelloController.java

```
package cn.tulingxueyuan.controllers;

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

/***
```

```
 * @Author 徐庶   QQ:1092002729
 * @Slogan 致敬大师，致敬未来的你
 */
@Controller
public class HelloController {

    @RequestMapping("/Hello")
    public String hello(String name){

        System.out.println("hello springmvc:"+name);

        return "redirect:index.jsp";
    }

}
```

5、流程细节

1、springmvc_helloworld运行流程：

通过上述的代码，我们能够总结出具体的运行流程：

1、客户端发送请求http://localhost:8080/springmvc01/hello

2、由tomcat接受到对应的请求

3、SpringMVC的前端控制器DispatcherServlet接收到所有的请求

4、查看请求地址和@RequestMapping注解的哪个匹配，来找到具体的类的处理方法

5、前端控制器找到目标处理类和方法之后，执行目标方法

6、方法执行完成之后会有一个返回值，SpringMVC会将这个返回值用视图解析器进行解析拼接成完整的页面地址

7、DispatcherServlet拿到页面地址之后，转发到具体的页面

2、springmvc的配置文件

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
         version="4.0">
    <!--配置DispatcherServlet-->
    <servlet>
        <servlet-name>springmvc</servlet-name>
        <servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <!--
           关联springmvc的配置文件
           此配置文件的属性可以不添加，但是需要在WEB-INF的目录下创建 前端控制器名称-servlet.xml文件
           -->
        <init-param>
```

```
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:applicationContext.xml</param-value>
        </init-param>
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>springmvc</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
</web-app>
```

3、DispatcherServlet的url-pattern

```

    匹配servlet的请求，
    /：标识匹配所有请求，但是不会jsp页面
    /*：拦截所有请求，拦截jsp页面

     但是需要注意的是，当配置成index.html的时候，会发现请求不到
     原因在于，tomcat下也有一个web.xml文件，所有的项目下web.xml文件都需要继承此web.xml
     在服务器的web.xml文件中有一个DefaultServlet用来处理静态资源，但是url-pattern是/
     而我们在自己的配置文件中如果添加了url-pattern=/会覆盖父类中的url-pattern，此时在请求的时候
     DispatcherServlet会去controller中做匹配，找不到则直接报404
     而在服务器的web.xml文件中包含了一个JspServlet的处理，所以不过拦截jsp请求
```