一、概述

> 一种轻量级的、基于MVC的Web层应用框架。偏前端而不是基于业务逻辑层；
>
> 作用：
>
> 1） 天生与Spring框架集成，如：(IOC,AOP)
>
> 2） 支持Restful风格
>
> 3） 进行更简洁的Web层开发
>
> 4） 支持灵活的URL到页面控制器的映射
>
> 5） 非常容易与其他视图技术集成，如:Velocity、FreeMarker等等
>
> 6） 因为模型数据不存放在特定的API里，而是放在一个Model里(Map数据结构实现，因此很容易被其他框架使用)
>
> 7） 非常灵活的数据验证、格式化和数据绑定机制、能使用任何对象进行数据绑定，不必实现特定框架的API
>
> 8） 更加简单、强大的异常处理
>
> 9） 对静态资源的支持
>
> 10） 支持灵活的本地化、主题等解析

主要的常用组件

- DispatcherServlet：前端控制器 / 核心控制器；
- Controller：处理器/页面控制器，做的是MVC中的C的事情，但控制逻辑转移到前端控制器了，用于对请求进行处理；
- HandlerMapping：请求映射到处理器，找谁来处理，如果映射成功返回一个HandlerExecutionChain对象（包含一个Handler处理器(页面控制器)对象、多个HandlerInterceptor拦截器对象）；
- View Resolver : 视图解析器，找谁来处理返回的页面。把逻辑视图解析为具体的View,进行这种策略模式，很容易更换其他视图技术；
    - 如：InternalResourceViewResolver将逻辑视图名映射为JSP视图；
- LocalResolver：本地化、国际化；
- MultipartResolver：文件上传解析器；
- HandlerExceptionResolver：异常处理器；

二、Helloworld

2.1 新建web工程，导入jar包

> 配置Tomcat服务器，在WEB-INF下创建lib目录存放jar包

```
spring-aop-4.0.0.RELEASE.jar
spring-beans-4.0.0.RELEASE.jar
spring-context-4.0.0.RELEASE.jar
spring-core-4.0.0.RELEASE.jar
spring-expression-4.0.0.RELEASE.jar
commons-logging-1.1.3.jar

spring-web-4.0.0.RELEASE.jar
spring-webmvc-4.0.0.RELEASE.jar
```

2.2 在web.xml中配置核心控制器

```
 <!--  配置前端控制器/核心控制器  -->
<servlet>
        <servlet-name>dispatcherServlet</servlet-name>
        <servlet-
class>org.springframework.web.servlet.DispatcherServlet</servlet-class>
        <!--    指定该Servlet初始化加载的IOC容器    -->
        <init-param>
            <param-name>contextConfigLocation</param-name>
            <param-value>classpath:spmvc-helloworld-config.xml</param-value>
        </init-param>
        <!--
             load-on-startup: 设置DispatcherServlet在服务器启动时加载。
             Servlet的创建时机:
                                1. 请求到达以后创建
                                2. 服务器启动即创建
         -->
        <load-on-startup>1</load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name>dispatcherServlet</servlet-name>
        <url-pattern>/</url-pattern>
</servlet-mapping>
```

2.3 配置IOC容器

spmvc-helloworld-config.xml

```
    <!--  组件扫描  -->
    <context:component-scan base-package="com.hello_spmvc"/>

    <!--
        配置视图解析器InternalResourceViewResolver

                工作机制:  prefix + 请求处理方法的返回值 + suffix  =  物理视图路径.
                /WEB-INF/views/success.jsp

        WEB-INF: 是服务器内部路径。 不能直接从浏览器端访问该路径下的资源. 但是可以内部转发进
行访问
    -->
```

```
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <!--    拼接物理视图路径      -->
        <property name="suffix" value=".jsp"></property>
        <property name="prefix" value="/WEB-INF/views/"></property>
    </bean>
```

2.4 编写处理请求的处理器，并标识为处理器

> 此时必须声明为@Controller组件

```
@Controller
public class HelloWorldHandle {
    /**
     * 映射请求的名称：用于客户端请求：类似Struts2中action映射配置的action名称
     * 1. 使用 @RequestMapping 注解来映射请求的 URL
     * 2. 返回值会通过视图解析器解析为实际的物理视图, 对于 InternalResourceViewResolver
视图解析器,
     * 会做如下的解析:
     *                通过 prefix + returnVal + suffix 这样的方式得到实际的物理视图, 然
后做转发操作.
     *                /WEB-INF/views/success.jsp
     */
    @RequestMapping("hello")
    public String helloWorld(){
        System.out.println("run helloWorld handle...");

        return "success";
    }
}
```

2.5 添加测试jsp文件

index.jsp（首页）

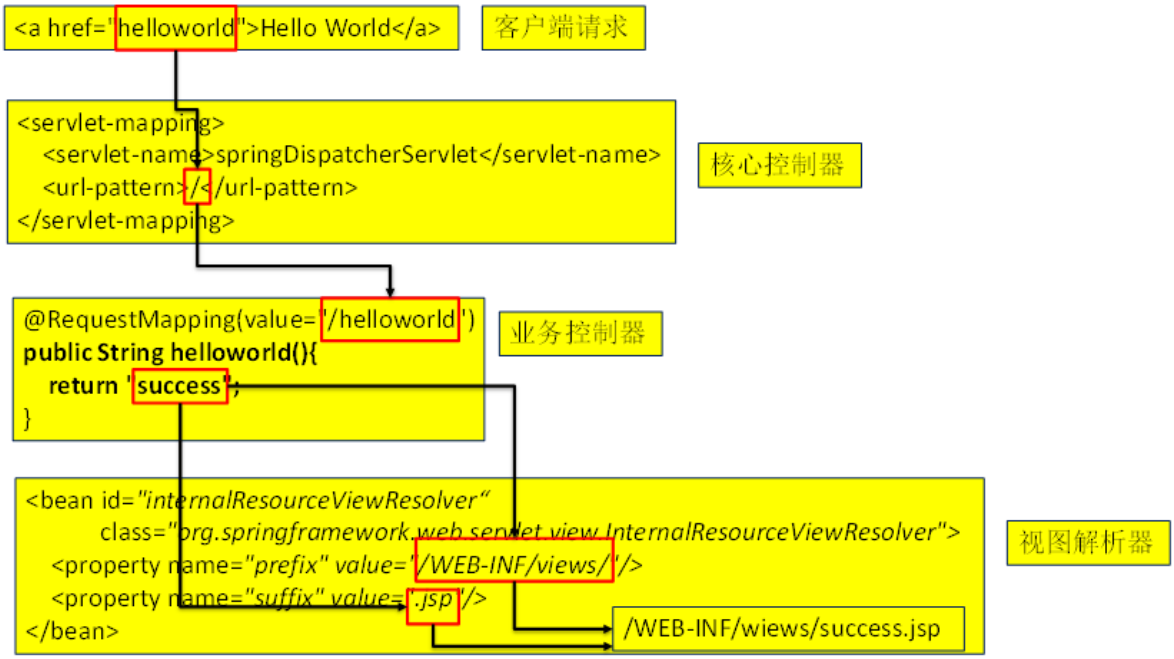
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
  <head>
    <title>Heelo spmvc</title>
  </head>
  <body>
    <!--
        在2.4请求处理器中接受该请求后，跳转到当前工程的hello.jsp页面

            1.注解@RequestMapping("hello")标识的方法返回"success"字符串
            2.访问视图解析器配置路径：/WEB-INF/views/success.jsp
    -->
    <a href="hello">跳转hello.jsp</a>
  </body>
</html>
```

success.jsp （要跳转的jsp）

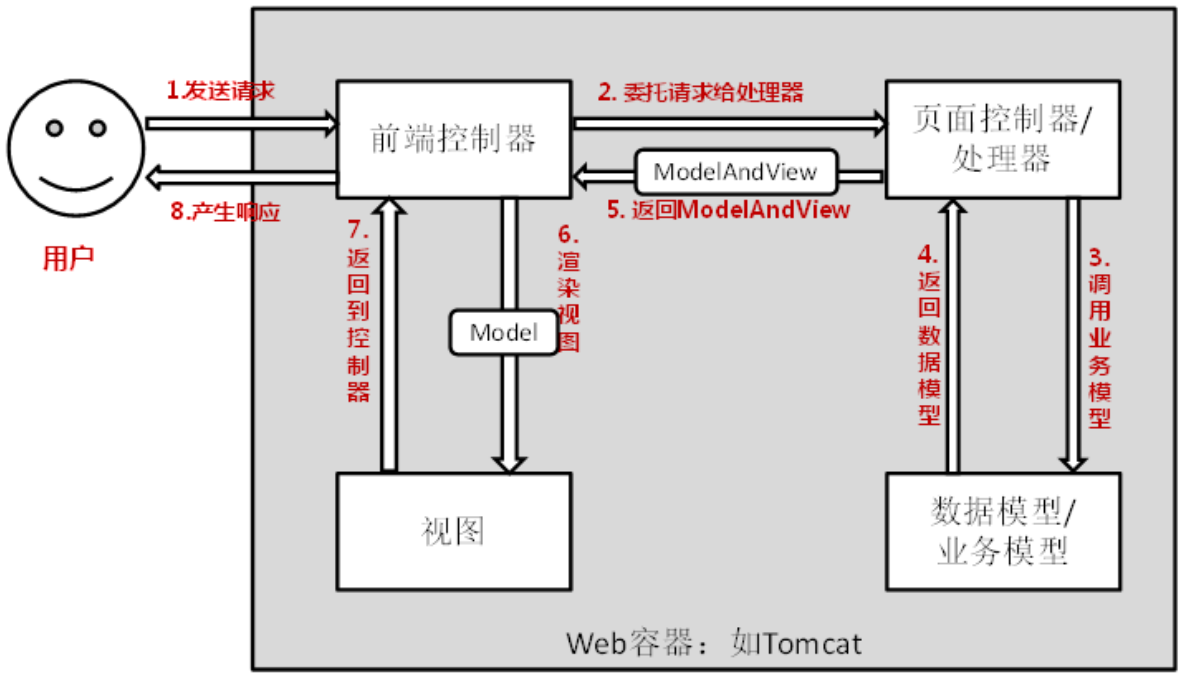
```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>Success Page</title>
</head>
<body>
    <h3>Success Page...</h3>
</body>
</html>
```

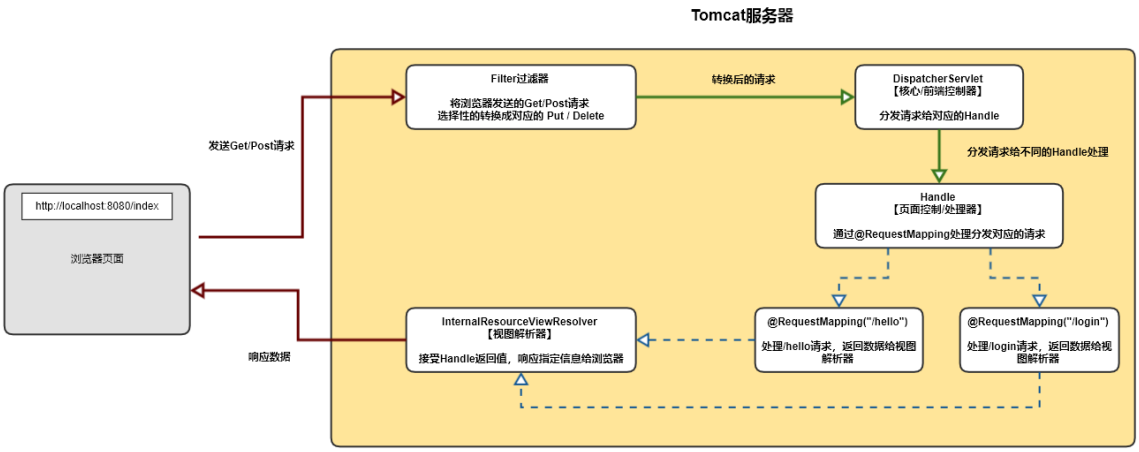
2.6 启动Tomcat服务器，测试跳转功能

请求流程示意图



流程分析示意图





三、@RequestMapping

四、REST