Spring MVC 各种场景下的使用

注解开发使用SpringMVC

一、我的第一个 SpringMVC 程序

之前用的是原生接口加+DispatcherServlet+映射器+适配器+视图解析器等创建了MVC程序，这次就用注解完成一个MVC程序的开发。

(0) 导入spring-webmv依赖

```
   <dependency>
      <groupId>junit</groupId>
      <artifactId>junit</artifactId>
      <version>4.11</version>
      <scope>test</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/javax.servlet/javax.servlet-api
-->
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>javax.servlet-api</artifactId>
      <version>3.1.0</version>
      <scope>provided</scope>
    </dependency>

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-
webmvc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.3.21</version>
    </dependency>
```

(1)在web.xml文件中配置信息

必须配置 DispatcherServlet，绑定spring-mvc.xml 配置文件，设置启动级别。

```
<?xml version="1.0" encoding="UTF-8" ?>
```

```xml
<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-app_4_0.xsd"
         version="4.0">

  <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:spring-mvc.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>
```

(2)在spring-mvc.xml 中配置MVC配置信息

在后面使用@Controller可以替代映射器和适配器的工作，不需要注册bean 使得id与class完成映射。如果返回的是JSON数据或者字符串，那么同样也不需要配置视图解析器了。在这里我们要经过jsp渲染视图所以需要配置视图解析器。

使用注解得配置以下信息：

> 开启注解扫描交给Spring容器管理

> 导入mvc的xml约束，开启mvc注解驱动

> 开启mvc静态资源处理器

```
为什么要过滤静态资源？

   SpringMVC 将接收到的所有请求都会被看做是一个普通的请求，包括静态资源的请求。这样一来，所有对于静态资源的请求都会被看作是一个普通的后台控制器请求，导致请求根本找不到从而报404错误

```

```
mvc 注解驱动的作用?

    支持注解驱动，如果不使用<mvc:annotation-driven/> 的话，要使得注解生效必须在上下文中注册 DefaultAnnotationHandlerMapping 和 AnnotationMethodHandlerAdapter 实例，这两个实例分别在类和方法级别进行处理，而annotation-driven 配置使得帮助我们自动完成上述实例的注入
```

```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"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans.xsd
        http://www.springframework.org/schema/context
        https://www.springframework.org/schema/context/spring-context.xsd
        http://www.springframework.org/schema/mvc
        http://www.springframework.org/schema/mvc/spring-mvc.xsd">


    <!--    视图解析器:  如果返回数据（JSON、HTML），不需要经过该步骤-->
    <!--    如果返回的是视图的话，那么原来的视图经过 之前的数据进行渲染之后 返回给Dispatcher，在展示给前端-->
    <bean class="org.springframework.web.servlet.view.InternalResourceViewResolver" id="internalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/jsp/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

<!--    开启注解扫描，将使用注解的类托管到spring 容器中-->
    <context:component-scan base-package="com.*"/>

<!--    过滤静态资源，  /.jsp  /.html 不会经过-->
    <mvc:default-servlet-handler/>

<!--    开启mvc注解驱动-->
    <mvc:annotation-driven/>

</beans>
```

(3) 写jsp文件

jsp文件等后端控制器返回数据后，渲染页面之后交给DispatcherServlet ,最后展示给前端。

```jsp
<%--
  Created by IntelliJ IDEA.
  User: rain7
  Date: 2022/6/24
  Time: 19:50
  To change this template use File | Settings | File Templates.
--%>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>111</title>
```

```
</head>
<body>

${msg}

</body>
</html>
```

(4) 写一个后端控制器

@Controller 相当于 映射器+适配器

@RequestMapper 反应了路由映射与具体的控制器的关系，默认情况下返回的是一个页面

@ResponseBody 如果这个注解与@RequestMapper搭配只用的话，那么规定返回的只能是一个非页面的数据

```
package com.bit.controller;

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

@Controller
public class HelloController{
   @RequestMapping("hello")
    public Object hello(Model model){
        // 封装数据
       model.addAttribute("msg","Hello SpringMVC");
       // 返回视图
       return "hello";
       // 将模型视图返回给视图解析器，找到 WEB-INF/jsp/hello.jsp 进行渲染视图
    }
}
```

(5) 部署项目，浏览器进行访问