

1.SpringMvc简介

1.1.什么是Spring MVC？简单介绍下你对Spring MVC的理解？

Spring MVC是一个基于Java的实现了MVC设计模式的请求驱动类型的轻量级Web框架，通过把模型-视图-控制器分离，将web层进行职责解耦，把复杂的web应用分成逻辑清晰的几部分，简化开发，减少出错，方便组内开发人员之间的配合。

1.2.Spring MVC的优点

1) 可以支持各种视图技术,而不仅仅局限于JSP； 2) 与Spring框架集成（如IoC容器、AOP等）； 3) 清晰的角色分配：前端控制器(dispatcherServlet)，请求到处理器映射(handlerMapping), 处理器适配器(HandlerAdapter), 视图解析器(ViewResolver)。 4) 支持各种请求资源的映射策略。

2.SpringMvc核心组件

2.1.Spring MVC的主要组件？

(1) **前端控制器 DispatcherServlet** (不需要程序员开发) 作用：接收请求、响应结果，相当于转发器，有了DispatcherServlet就减少了其它组件之间的耦合度。

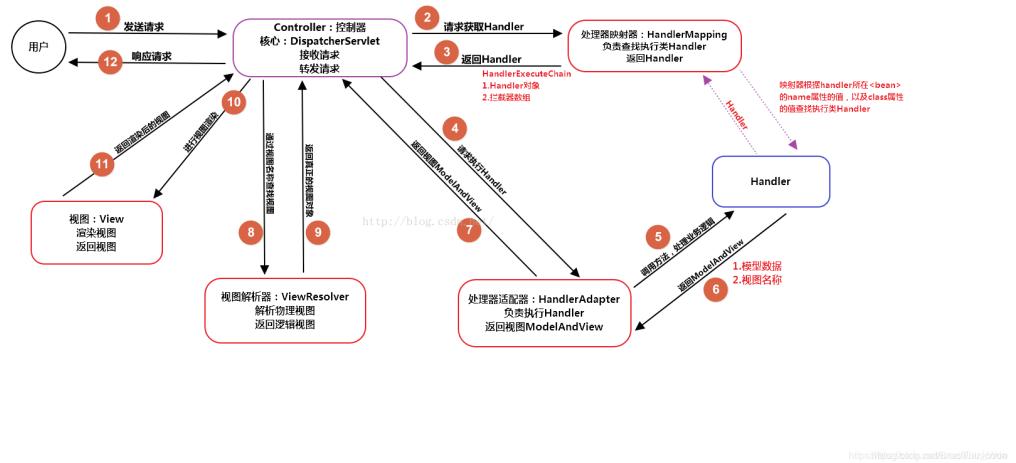
(2) **处理器映射器HandlerMapping** (不需要程序员开发) 作用：根据请求的URL来查找Handler

(3) **处理器适配器HandlerAdapter** 注意：在编写Handler的时候要按照HandlerAdapter要求的规则去编写，这样适配器HandlerAdapter才可以正确的去执行Handler。

(4) **处理器Handler (需要程序员开发)**

(5) **视图解析器 ViewResolver (不需要程序员开发)** 作用：进行视图的解析，根据视图逻辑名解析成真正的视图 (view)

(6) 视图View (需要程序员开发jsp) View是一个接口，它的实现类支持不同的视图类型 (jsp, freemarker, pdf等等)



2.2.什么是DispatcherServlet

Spring的MVC框架是围绕DispatcherServlet来设计的，它用来处理所有的HTTP请求和响应。

2.3.什么是Spring MVC框架的控制器？

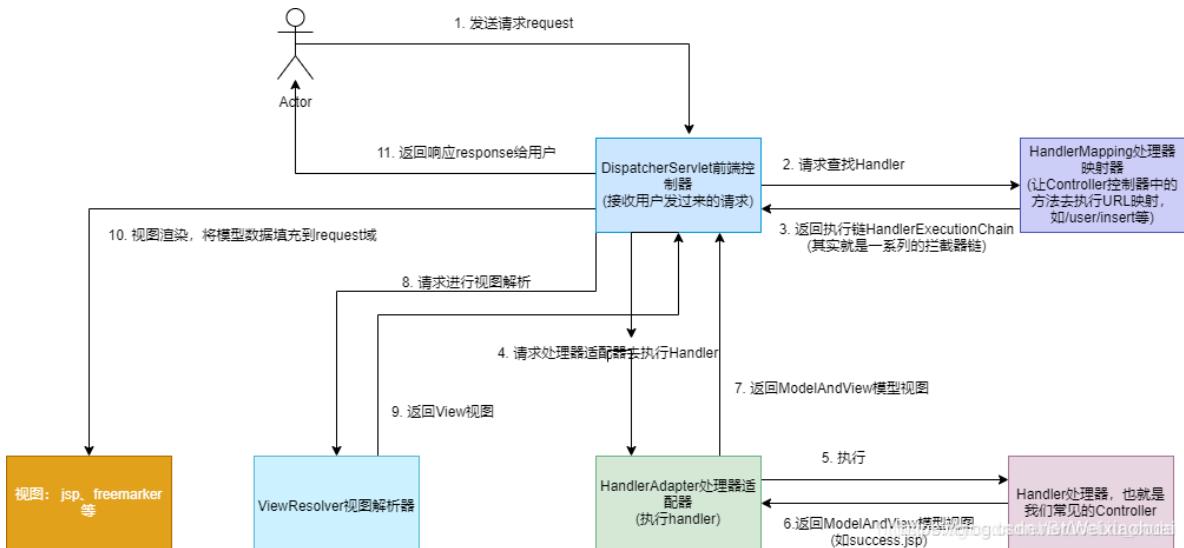
控制器提供一个访问应用程序的行为，此行为通常通过服务接口实现。控制器解析用户输入并将其转换为一个由视图呈现给用户的模型。Spring用一个非常抽象的方式实现了一个控制层，允许用户创建多种用途的控制器。

2.4.Spring MVC的控制器是不是单例模式,如果是,有什么问题,怎么解决?

是单例模式,所以在多线程访问的时候有线程安全问题,不要用同步,会影响性能的,解决方案是在控制器里面不能写字段。

3. 工作原理

3.1.请描述Spring MVC的工作流程？描述一下 DispatcherServlet 的工作流程？



- (1) 用户发送请求至前端控制器DispatcherServlet; (2) DispatcherServlet收到请求后，调用HandlerMapping处理器映射器，请求获取Handle； (3) 处理器映射器根据请求url找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给DispatcherServlet； (4) DispatcherServlet 调用 HandlerAdapter处理器适配器； (5) HandlerAdapter 经过适配调用 具体处理器(Handler，也叫后端控制器)； (6) Handler执行完成返回ModelAndView； (7) HandlerAdapter将Handler执行结果 ModelAndView返回给DispatcherServlet； (8) DispatcherServlet将 ModelAndView传给ViewResolver视图解析器进行解析； (9) ViewResolver解析后返回具体View； (10) DispatcherServlet对View进行渲染视图（即将模型数据填充至视图中） (11) DispatcherServlet响应用户。

4.MVC框架

4.1.MVC是什么？MVC设计模式的好处有哪些

mvc是一种设计模式（设计模式就是日常开发中编写代码的一种好的方法和经验的总结）。模型（model）-视图（view）-控制器（controller），三层架构的设计模式。用于实现前端页面的展现与后端业务数据处理的分离。

mvc设计模式的好处 1.分层设计，实现了业务系统各个组件之间的解耦，有利于业务系统的可扩展性，可维护性。 2.有利于系统的并行开发，提升开发效率。

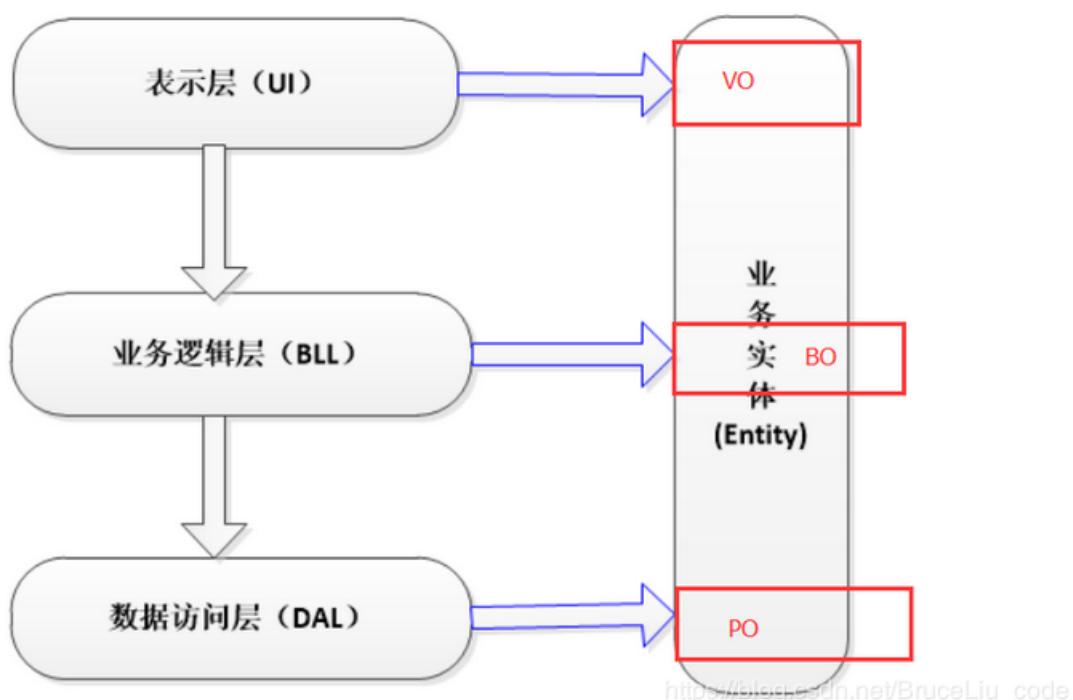
4.2.三层架构

我们的开发架构一般都是基于两种形式，一种是 C/S 架构，也就是客户端/服务器，另一种是 B/S 架构，也就是浏览器服务器。在 JavaEE 开发中，几乎全都是基于 B/S 架构的开发。那么在 B/S 架构中，系统标准的三层架构包括：表现层、业务层、持久层。三层架构在我们的实际开发中使用的非常多，所以我们课程中的案例也都是基于三层架构设计的。三层架构中，每一层各司其职，接下来我们就说说每层都负责哪些方面：

- 表现层：也就是我们常说的web层。它负责接收客户端请求，向客户端响应结果，通常客户端使用http协议请求web 层， web 需要接收 http 请求，完成 http

响应。表现层包括展示层和控制层：控制层负责接收请求，展示层负责结果的展示。表现层依赖业务层，接收到客户端请求一般会调用业务层进行业务处理，并将处理结果响应给客户端。表现层的设计一般都使用 MVC 模型。（MVC 是表现层的设计模型，和其他层没有关系）

- 业务层：也就是我们常说的 service 层。它负责业务逻辑处理，和我们开发项目的需求息息相关。web 层依赖业务层，但是业务层不依赖 web 层。业务层在业务处理时可能会依赖持久层，如果要对数据持久化需要保证事务一致性。（也就是我们说的，事务应该放到业务层来控制）
- 持久层：也就是我们常说的 dao 层。负责数据持久化，包括数据层即数据库和数据访问层，数据库是对数据进行持久化的载体，数据访问层是业务层和持久层交互的接口，业务层需要通过数据访问层将数据持久化到数据库中。通俗的讲，持久层就是和数据库交互，对数据库表进行增删改查的。

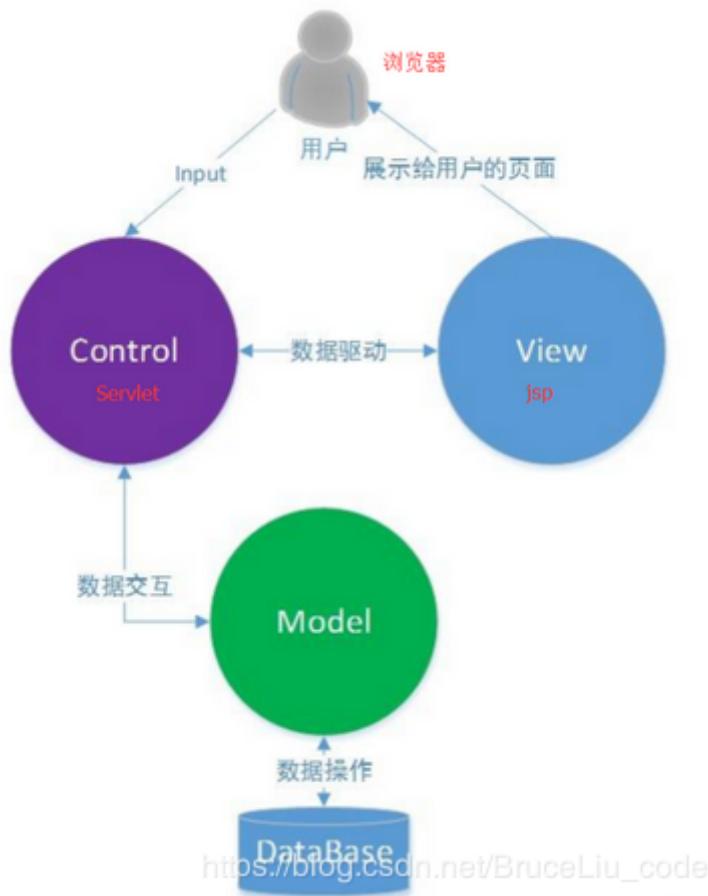


4.3. MVC 模型

MVC 全名是 Model View Controller，是模型(model) - 视图(view) - 控制器(controller)的缩写，是一种用于设计创建 Web 应用程序表现层的模式。MVC 中每个部分各司其职：

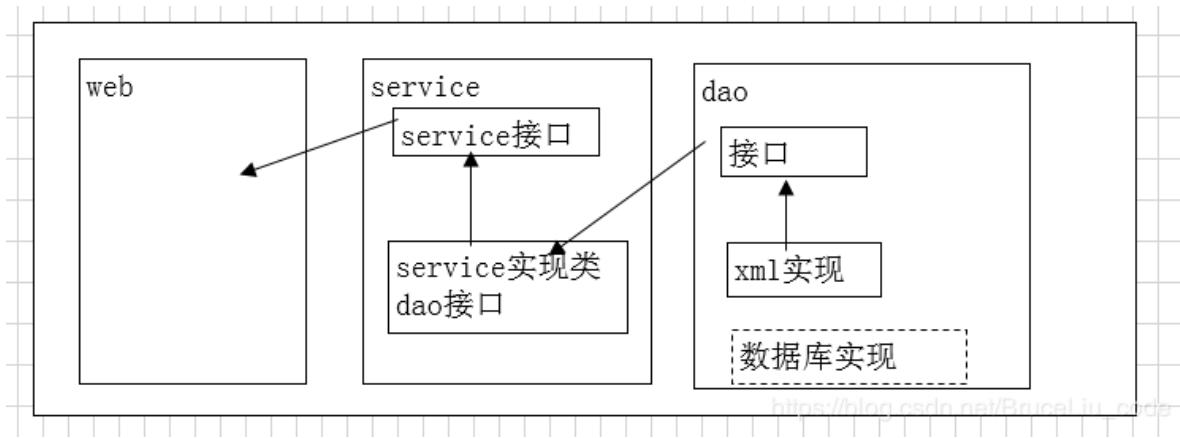
- Model (模型)：通常指的就是我们的数据模型。作用一般情况下用于封装数据。
- View (视图)：通常指的就是我们的 jsp 或者 html。作用一般就是展示数据的。通常视图是依据模型数据创建的。
- Controller (控制器)：是应用程序中处理用户交互的部分。作用一般就是处理程序逻辑的。它相对于前两个不是很好理解，这里举个例子：例如：我们要保存一个用户的信息，该用户信息中包含了姓名，性别，年龄等等。这时候表单输入要求年龄必须是 1~100 之间的整数。姓名和性别不能为空。并且把数据填充

到模型之中。此时除了js的校验之外，服务器端也应该有数据准确性的校验，那么校验就是控制器的该做的。当校验失败后，由控制器负责把错误页面展示给使用者。如果校验成功，也是控制器负责把数据填充到模型，并且调用业务层实现完整的业务需求。



4.4.经典三层架构和MVC的关系

他们是两个毫无相关的东西，经典三层架构是一种分层思想，将开发模式分为了这三层，每个人根据自己的专长，开发不同的模块，比如，前端工程师，那么就专研表示层即可，想办法如何让页面变的更好看，如何吸引别人，而有些专门做数据库工作的人，就可以只关注操作数据库的活，如何让查询更加快速有效，而不必关注数据该如何显示这种问题。这就是分层带来的巨大好处。而MVC是一种设计模式，目的是让HTML代码和业务逻辑代码分开，让代码看起来更加清晰，便于开发。硬说他们有关系的话，只能说他们有共同的点，分层，解耦。实际项目中的包命名结构，其也是按照三层架构思想来进行编写代码的，脑袋里要保持着这种思想进行开发。



xxx:代表公司名称 yyy: 代表项目名称

| | | |
|------------------------|-------------------------------------|-----------------------|
| com.xxx.yyy.dao | dao层接口 com.xxx.yyy.dao.impl | dao层实现 |
| com.xxx.yyy.service | service层接口 com.xxx.yyy.service.impl | service层实 |
| 现 | com.xxx.yyy.web | web层 com.xxx.yyy.util |
| 工具包 com.xxx.yyy.domain | | javabean |

5.常用注解

5.1.注解原理是什么

注解本质是一个继承了Annotation的特殊接口，其具体实现类是Java运行时生成的动态代理类。我们通过反射获取注解时，返回的是Java运行时生成的动态代理对象。通过代理对象调用自定义注解的方法，会最终调用AnnotationInvocationHandler的invoke方法。该方法会从memberValues这个Map中索引出对应的值。而memberValues的来源是Java常量池。

5.2.Spring MVC常用的注解有哪些？

@RequestMapping：用于处理请求 url 映射的注解，可用于类或方法上。用于类上，则表示类中的所有响应请求的方法都是以该地址作为父路径。 @RequestBody：注解实现接收http请求的json数据，将json转换为java对象。 @ResponseBody：注解实现将controller方法返回对象转化为json对象响应给客户。

5.3.SpringMvc中的控制器的注解一般用哪个,有没有别的注解可以替代？

一般用@Controller注解,也可以使用@RestController,@RestController注解相当于@ResponseBody + @Controller,表示是表现层,除此之外，一般不用别的注解代替。

5.4.@Controller注解的作用

在Spring MVC 中，控制器Controller 负责处理由DispatcherServlet 分发的请求，它把用户请求的数据经过业务处理层处理之后封装成一个Model，然后再把该Model 返回给对应的View 进行展示。在Spring MVC 中提供了一个非常简便的定义Controller 的方法，你无需继承特定的类或实现特定的接口，只需使用@Controller 标记一个类是

Controller，然后使用@RequestMapping 和@RequestParam 等一些注解用以定义URL 请求和Controller 方法之间的映射，这样的Controller 就能被外界访问到。此外 Controller 不会直接依赖于HttpServletRequest 和HttpServletResponse 等HttpServlet 对象，它们可以通过Controller 的方法参数灵活的获取到。

@Controller 用于标记在一个类上，使用它标记的类就是一个Spring MVC Controller 对象。分发处理器将会扫描使用了该注解的类的方法，并检测该方法是否使用了 @RequestMapping 注解。@Controller 只是定义了一个控制器类，而使用 @RequestMapping 注解的方法才是真正处理请求的处理器。单单使用@Controller 标记在一个类上还不能真正意义上的说它就是Spring MVC 的一个控制器类，因为这个时候Spring 还不认识它。那么要如何做Spring 才能认识它呢？这个时候就需要我们把这个控制器类交给Spring 来管理。有两种方式：

- 1 在Spring MVC 的配置文件中定义MyController 的bean 对象。
- 2 在Spring MVC 的配置文件中告诉Spring 该到哪里去找标记为@Controller 的 Controller 控制器。

5.5.@RequestMapping注解的作用

RequestMapping是一个用来处理请求地址映射的注解，可用于类或方法上。用于类上，表示类中的所有响应请求的方法都是以该地址作为父路径。

RequestMapping注解有六个属性，下面我们把她分成三类进行说明（下面有相应示例）。value, method

- 1 value: 指定请求的实际地址，指定的地址可以是URI Template 模式（后面将会说明）；
- 2 method: 指定请求的method类型， GET、POST、PUT、DELETE等；

consumes, produces

- 1 consumes: 指定处理请求的提交内容类型（Content-Type），例如 application/json, text/html；
- 2 produces: 指定返回的内容类型，仅当request请求头中的(Accept)类型中包含该指定类型才返回；

params, headers

- 1 params: 指定request中必须包含某些参数值是，才让该方法处理。
- 2 headers: 指定request中必须包含某些指定的header值，才能让该方法处理请求。

5.6.@ResponseBody注解的作用

作用：该注解用于将Controller的方法返回的对象，通过适当的HttpMessageConverter转换为指定格式后，写入到Response对象的body数据区。

使用时机：返回的数据不是html标签的页面，而是其他某种格式的数据时（如json、xml等）使用；

5.7.@PathVariable和@RequestParam的区别

请求路径上有个id的变量值，可以通过@PathVariable来获取 @RequestMapping(value = “/page/{id}”, method = RequestMethod.GET)

@RequestParam用来获得静态的URL请求入参 spring注解时action里用到。

6.其他

6.1.Spring MVC与Struts2区别

相同点

- 1 都是基于mvc的表现层框架，都用于web项目的开发。

不同点

1. 前端控制器不一样。Spring MVC的前端控制器是servlet: DispatcherServlet。struts2的前端控制器是filter: StrutsPreparedAndExcutorFilter。
- 2
3. 请求参数的接收方式不一样。Spring MVC是使用方法的形参接收请求的参数，基于方法的开发，线程安全，可以设计为单例或者多例的开发，推荐使用单例模式的开发（执行效率更高），默认就是单例开发模式。struts2是通过类的成员变量接收请求的参数，是基于类的开发，线程不安全，只能设计为多例的开发。
- 4
5. Struts采用值栈存储请求和响应的数据，通过OGNL存取数据，Spring MVC通过参数解析器是将request请求内容解析，并给方法形参赋值，将数据和视图封装成 ModelAndView对象，最后又将 ModelAndView中的模型数据通过request域传输到页面。Jsp视图解析器默认使用jstl。
- 6
7. 与spring整合不一样。Spring MVC是spring框架的一部分，不需要整合。在企业项目中，Spring MVC使用更多一些。

6.2.Spring MVC怎么样设定重定向和转发的？

- 1 (1) 转发：在返回值前面加“forward：“，譬如“forward:user.do?name=method4”
- 2 (2) 重定向：在返回值前面加“redirect：“，譬如“redirect:http://www.baidu.com”

6.3.Spring MVC怎么和AJAX相互调用的？

通过Jackson框架就可以把Java里面的对象直接转化成Js可以识别的Json对象。具体步骤如下：

```
1 (1) 加入Jackson.jar  
2 (2) 在配置文件中配置json的映射  
3 (3) 在接受Ajax方法里面可以直接返回Object, List等, 但方法前面要加上  
@ResponseBody注解。
```

6.4.如何解决POST请求中文乱码问题, GET的又如何处理呢?

(1) 解决post请求乱码问题: 在web.xml中配置一个CharacterEncodingFilter过滤器, 设置成utf-8;

```
1 <filter>  
2     <filter-name>CharacterEncodingFilter</filter-name>  
3     <filter->  
4         <class>org.springframework.web.filter.CharacterEncodingFilter</filter-class>  
5     </filter->  
6     <init-param>  
7         <param-name>encoding</param-name>  
8         <param-value>utf-8</param-value>  
9     </init-param>  
10    </filter>  
11    <filter-mapping>  
12        <filter-name>CharacterEncodingFilter</filter-name>  
13        <url-pattern>/*</url-pattern>  
14    </filter-mapping>
```

(2) get请求中文参数出现乱码解决方法有两个: ①修改tomcat配置文件添加编码与工程编码一致, 如下:

```
1 <ConnectorURIEncoding="utf-8" connectionTimeout="20000"  
port="8080" protocol="HTTP/1.1" redirectPort="8443" />
```

②另外一种方法对参数进行重新编码:

```
1 String userName = new  
String(request.getParamter("userName").getBytes("ISO8859-1"), "utf-  
8")
```

ISO8859-1是tomcat默认编码, 需要将tomcat编码后的内容按utf-8编码。

6.5.怎样在方法里面得到Request,或者Session?

直接在方法的形参中声明request, Spring MVC就自动把request对象传入。

6.6.如果想在拦截的方法里面得到从前台传入的参数,怎么得到?

直接在形参里面声明这个参数就可以,但必须名字和传过来的参数一样。

6.7.如果前台有很多个参数传入,并且这些参数都是一个对象的,那么怎么样快速得到这个对象?

直接在方法中声明这个对象, Spring MVC就自动会把属性赋值到这个对象里面。

6.8.Spring MVC中函数的返回值是什么?

返回值可以有很多类型,有String, ModelAndView。 ModelAndView类把视图和数据都合并的一起的, 但一般用String比较好。

6.9.Spring MVC用什么对象从后台向前台传递数据的?

通过ModelMap对象,可以在这个对象里面调用put方法,把对象加到里面,前台就可以通过el表达式拿到。

6.10.Spring MVC里面拦截器是怎么写的

有两种写法,一种是实现HandlerInterceptor接口, 另外一种是继承适配器类, 接着在接口方法当中, 实现处理逻辑; 然后在Spring MVC的配置文件中配置拦截器即可:

```
1 <!-- 配置Spring MVC的拦截器 -->
2 <mvc:interceptors>
3     <!-- 配置一个拦截器的Bean就可以了 默认是对所有请求都拦截 -->
4     <bean id="myInterceptor"
5         class="com.zwp.action.MyHandlerInterceptor"></bean>
6     <!-- 只针对部分请求拦截 -->
7     <mvc:mapping path="/modelMap.do" />
8     <bean class="com.zwp.action.MyHandlerInterceptorAdapter"
9         />
10    </mvc:interceptor>
11 </mvc:interceptors>
```

