# Controller后端处理器常用注解

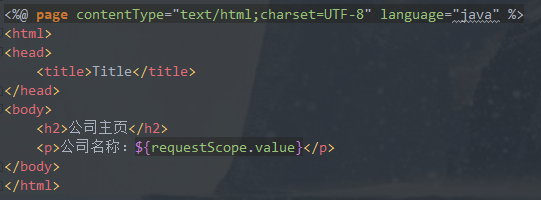
## controller的基本原理

### 应用背景

1. 在SpringMVC 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。
2. 在SpringMVC 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。
3. 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。

### 简单的例子





1. @Controller表示这个类是个后端处理器类，用来处理DispatcherServlet分发的请求。
2. @RequestMapping表示访问的地址
3. ModelAndView类的作用是将数据存进去了requestScope域对象中，将view视图名称交给了视图解析器。

### 注意点

1. request和requestScope的区别是什么？

**Request**对象位于servlet包中，可以获取请求传入的参数getParameter,也可以在request中设置获取值getAttribute.

**RequestScope**对象位于spring包中，存入Model对象的键值对存在了requestScope中，在JSP中通过requestScope获取，一般不用在controller中。

1. 在controller中怎么获取request和requestScope对象？

直接将HttpServletRequest ,HttpServletResponse 放在方法的形参上就可以获取request和respose对象。

requestScope对象一般不用在controller中。

1. request对象中getParameter和getAttribute区别？

request.getParameter() 取得是通过容器的实现来取得Post或者Get方法传入的数据，会从Web客户端传到Web服务器端，代表HTTP请求数据。

request.setAttribute()和getAttribute()只是在web容器内部流转，仅仅是请求处理阶段。

## @Controller的使用

### 2.1 简单的使用

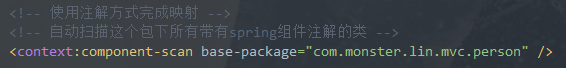


1. @Controller 用于标记在一个类上，使用它标记的类就是一个SpringMVC Controller 对象。分发处理器将会扫描使用了该注解的类的方法，并检测该方法是否使用了@RequestMapping 注解。@Controller 只是定义了一个控制器类，而使用@RequestMapping 注解的方法才是真正处理请求的处理器。
2. 使用@Controller 标记在一个类上还没将这个控制器类注册到spring中，还需要有一个注册控制器类（注册bean）的动作。

第一个方法就是在xml文件中注册bean：

<bean class="com.monster.lin.mvc.person.blog.controller.ViewerController"/>

第二个方法是开启注解功能，让spring自动扫描注解：



### 2.2 @Controller,@RestController和@ResposeBody的区别联系

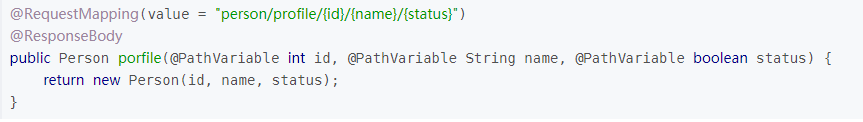
1. 在使用springmvc框架的时候，在处理json的时候需要用到spring框架特有的注解@ResponseBody或者@RestController注解，这两个注解都会处理返回的数据格式，使用了该类型注解后返回的不再是视图，不会进行转跳，而是返回json或xml数据格式，输出在页面上。
2. @ResponseBody，一般是使用在单独的方法上的，需要哪个方法返回json数据格式，就在哪个方法上使用，具有针对性。
3. @RestController，一般是使用在类上的，它表示的意思其实就是结合了@Controller和@ResponseBody两个注解，表示该类中所有的处理方法都是以@ResposeBody进行。

### 2.3 @ResponseBody 和 @RequestBody 注解的区别



1. @RequestBody 注解是将HTTP请求正文插入方法中，使用适合的 HttpMessageConverter 将请求体写入某个对象。
2. @Responsebody 注解表示该方法的返回的结果直接写入 HTTP 响应正文（ResponseBody）中，一般在异步获取数据时使用.
3. 在使用 @RequestMapping 后，返回值通常解析为跳转路径，加上 @Responsebody 后返回结果不会被解析为跳转路径，而是直接写入HTTP 响应正文中。例如，异步获取 json 数据，加上 @Responsebody 注解后，就会直接返回 json 数据。

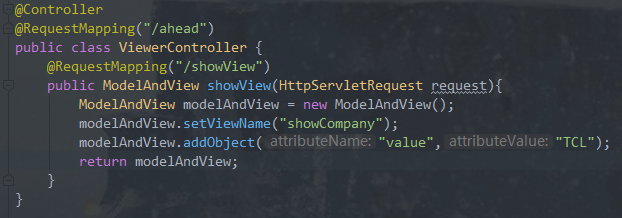
### 2.4 @PathVariable 注解



1. @PathVariable作用是获取请求中的参数，并将这个参数写入requestMapping的动态路径中。

## @RequestMapping注解的使用

### 基本用法



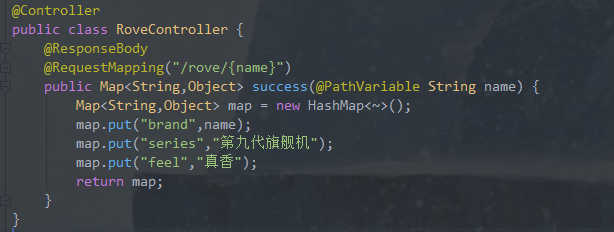
1. 当@RequestMapping 标记在Controller 类上的时候，里面使用@RequestMapping 标记的方法的请求地址都是相对于类上的@RequestMapping 而言的.

此时访问路径为：/项目名/ahead/showView

1. 当Controller 类上没有标记@RequestMapping 注解时，方法上的@RequestMapping 都是绝对路径。

此时访问路径为：/项目名/showView

### 注解@PathVariable-绑定URL参数到方法入参

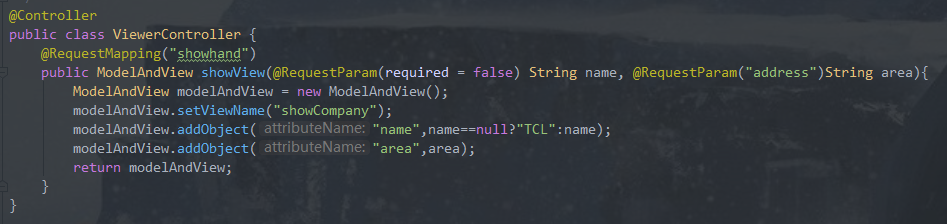


此时访问地址可以为：http://localhost:8999/blog/rove/XIAOMI

1. @PathVariable可以将URL中的参数值，应用到方法入参中。
2. 如果注解中不含变量名，则默认URL参数名和方法参数名相同才能赋值。如果名称不一致，则需要在注解中指定变量名。
3. 此外，RequestMapping可以使用通配符，如@RequestMapping(“\*/showView”).

实际访问地址为： http://localhost:8999/blog(项目名)/任意值/showView

### 注解@RequestParam绑定请求参数到方法参数



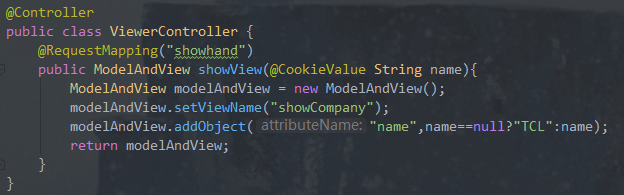
1. 在spring中，默认是将同名的请求参数和方法参数绑定在一起，而且请求参数中必须传入同名的方法参数，否则就会报错。
2. @RequestParam(required = false) String name

属性required表示，是否一定需要传入这个参数，默认为true。如果修改为false，则改参数可传可不传。

1. @RequestParam("address")String area

注解的变量名代表请求参数中的变量名，如果当名字不同时，可以通过这个方式将请求参数中不相同名字的变量绑定在方法参数中。

### 注解@CookieValue 绑定 cookie 的值到 Controller 方法参数



1. 作用是将HTTP请求中cookie的值绑定到方法参数中，默认按照参数名称匹配。也可以通过指定变量名，来匹配cookie值和方法参数。

### 注解@RequestHeader

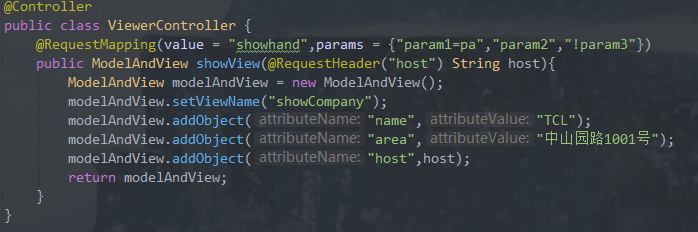
作用：绑定 HttpServletRequest 头信息到 Controller 方法参数



1. 注解@RequestHeader,@PathVariable,@RequestParam,@CookieValue的参数匹配规则是一样的，即默认按照参数名称匹配，也可以指定变量名匹配。
2. 特别的是，注解@RequestHeader对参数名称的大小写不敏感，而其他三个注解对大小写均敏感。

### 注解@RequestMapping的高级属性

* + 1. params属性



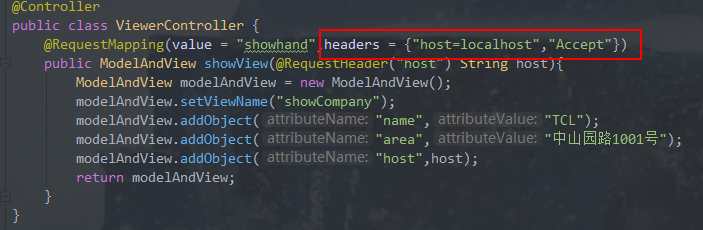
1. value表示访问路径。
2. params属性对**请求参数**做出了约束条件。

param1参数必须等于pa,param2参数必须存在，param3参数必须不存在。

* + 1. method属性



1. method属性是限制HTTP请求的方法类型，比如GET,POST,DELETE等等。
   * 1. headers属性



1. headers 属性可以通过请求头信息来缩小@RequestMapping 的映射范围。
2. headers 属性的用法和功能与params 属性相似。

### 以 @RequestMapping 标记的处理器方法 支持的方法参数和返回类型

* + 1. 支持的方法参数类型

1. **HttpServlet 对象,主要包括HttpServletRequest 、HttpServletResponse 和HttpSession 对象。** 这些参数**Spring 在调用处理器方法的时候会自动给它们赋值**，所以当在处理器方法中需要使用到这些对象的时候，可以直接在方法上给定一个方法参数的申明，然后在方法体里面直接用就可以了。但是有一点需要注意的是在使用HttpSession 对象的时候，如果此时HttpSession 对象还没有建立起来的话就会有问题。
2. Spring 自己的WebRequest 对象。 使用该对象可以访问到存放在HttpServletRequest 和HttpSession 中的属性值。
3. InputStream 、OutputStream 、Reader 和Writer 。

InputStream 和Reader 是针对HttpServletRequest 而言的，可以从里面取数据；

OutputStream 和Writer 是针对HttpServletResponse 而言的，可以往里面写数据。

1. 使用@PathVariable 、@RequestParam 、@CookieValue 和@RequestHeader 标记的参数。
2. 使用@ModelAttribute 标记的参数。
3. java.util.Map 、Spring 封装的Model 和ModelMap 。 这些都可以用来封装模型数据，用来给视图做展示。
4. 实体类。 可以用来接收上传的参数。
5. Spring 封装的MultipartFile 。 用来接收上传文件的。
6. Spring 封装的Errors 和BindingResult 对象。 这两个对象参数必须紧接在需要验证的实体对象参数之后，它里面包含了实体对象的验证结果。
   * 1. 支持的返回类型
7. 一个包含模型和视图的ModelAndView 对象。
8. 一个模型对象，这主要包括Spring 封装好的Model 和ModelMap ，以及java.util.Map ，当没有视图返回的时候视图名称将由RequestToViewNameTranslator 来决定。
9. 一个View 对象。这个时候如果在渲染视图的过程中模型的话就可以给处理器方法定义一个模型参数，然后在方法体里面往模型中添加值。
10. 一个String 字符串。这往往代表的是一个视图名称。这个时候如果需要在渲染视图的过程中需要模型的话就可以给处理器方法一个模型参数，然后在方法体里面往模型中添加值就可以了。
11. **如果处理器方法被注解@ResponseBody 标记的话，那么处理器方法的任何返回类型都会通过HttpMessageConverters 转换之后写到HttpServletResponse 中，而不会像上面的那些情况一样当做视图或者模型来处理。**
12. 返回值是void 。这种情况一般是我们直接把返回结果写到HttpServletResponse 中了，如果没有写的话，那么Spring 将会利用RequestToViewNameTranslator 来返回一个对应的视图名称。如果视图中需要模型的话，处理方法与返回字符串的情况相同。
13. 除以上几种情况之外的其他任何返回类型都会被当做模型中的一个属性来处理，而返回的视图还是由RequestToViewNameTranslator 来决定，添加到模型中的属性名称可以在该方法上用@ModelAttribute(“attributeName”) 来定义，否则将使用返回类型的类名称的首字母小写形式来表示。使用@ModelAttribute 标记的方法会在@RequestMapping 标记的方法执行之前执行。

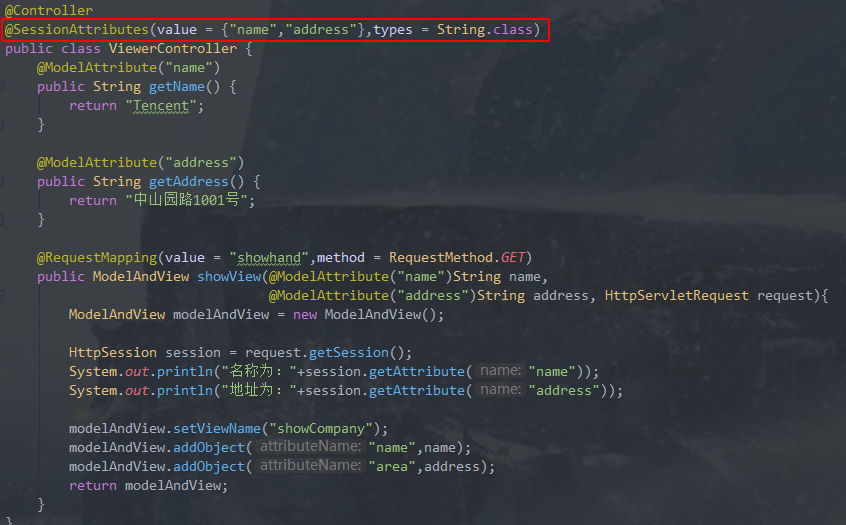
### 使用 @ModelAttribute 和 @SessionAttributes 传递和保存数据

作用：SpringMVC 支持使用 @ModelAttribute 和 @SessionAttributes 在不同的模型和控制器之间共享数据。

* + 1. @ModelAttribut模型属性



1. 当 @ModelAttribute 标记在方法上的时候，**该方法将在处理器方法执行之前执行**，然后把返回的对象存放在 session 或模型属性中，属性名称可以使用 @ModelAttribute(“attributeName”) 在标记方法的时候指定，若未指定，则使用返回类型的类名称（首字母小写）作为属性名称。
2. 设置属性的时候，@ModelAttribute用在方法上，使用的时候，用在@RequestMapping的方法参数上。
   * 1. @SessionAttribute属性



1. @SessionAttribute一般用在Controller类上，可以通过名称、类型或者名称加类型的形式来指定哪些属性是需要存放在session 中的。
2. @SessionAttributes添加的类型是，在Model中存在的类型，只有在模型中存在的键值对才能通过这个注解添加到session。添加到Model中有两个方法，一个就是注解@ModelAttributes，另一个就是获取ModelMap对象，往里面添加属性。
3. 注解session是在处理器方法执行后再添加到session中的，所以首次访问的时候，session一般为空，当第二次访问到时候，session就有值了。
4. @SessionAttributes可以通过value或者type添加，如果添加的属性有重叠则取并集。

### 自定义类型转换器

作用：**请求参数根据自定义规则转换为方法参数。**在通过处理器方法参数接收 request 请求参数绑定数据的时候，对于一些简单的数据类型 Spring 会帮我们自动进行类型转换，而对于一些复杂的类型由于 Spring 没法识别，

* + 1. 通过@ InitBinder注册（只对Controller类有效）
* 注册的基本步骤：

1. @InitBinder注解声明注册方法，并且方法参数为注册器对象。
2. WebDateBinder为注册器对象,registerCustomEditor为注册方法。
3. 初始化自定义编辑器对象PropertyEditor,系统自带一些简单的编辑器对象，如

CustomDateEditor.CustomoBooleanEditor.

1. 注册编辑器对象，第一个参数为目前类型对象，第二个参数为自定义的编辑器。

（系统提供的编辑器对象）



需要用@PathVarible,@RequestParam等注解指定请求参数和方法入参匹配，否则会发生类型不匹配错误。只有指定了之后，才可以调用类型转换器进行类型的转换。

（自定义编辑器）



实现自定义编辑器

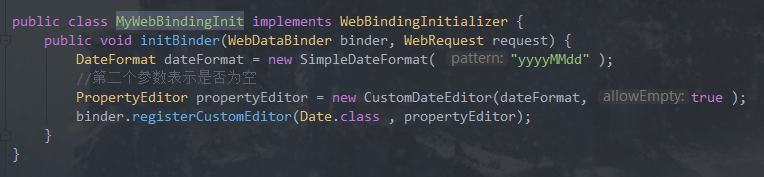
1.通过PropertyEditorSupport实例化PropertyEditor对象

2.重写setAsText方法和getAsText方法

* + 1. 实现 WebBindingInitializer 接口定义全局的类型转换器
* 基本步骤：

1. 新建一个类实现WebBindingInitializer接口，重写InitBinder方法。
2. 将该类加入springBean中，利用自己的类覆盖AnnotationMethodHandlerAdapter类型中的属性webBindingInitializer。

（新建类实现接口WebBindingInitializer）



（将新建类覆盖类RequestMappingHandlerAdapter属性webBindingInitializer）



AnnotationMethodHandlerAdapter已经过时，换成RequestMappingHandlerAdapter

3.9.3 触发类型转换器

1. 当Controller处理器方法参数使用@RequestParam、@PathVariable、@RequestHeader、@CookieValue和@ModelAttribute标记的时候都会触发initBinder方法的执行，这包括使用WebBindingInitializer定义的全局方法和在Controller中使用@InitBinder标记的局部方法。
2. 每个使用了这几个注解标记的参数都会触发一次initBinder方法的执行，这也意味着有几个参数使用了上述注解就会触发几次initBinder方法的执行。

### 注解@Value读取配置文件内容



1. 注解@Value可以将括号中的字符串赋值到属性上