spring mvc常用的注解：

个介绍。

@Controller

@Controller 负责注册一个bean 到spring 上下文中，bean 的ID 默认为

类名称开头字母小写,你也可以自己指定，如下

方法一：

@Controller

public class TestController {}

方法二：

@Controller("tmpController")

public class TestController {}

@RequestMapping

1.@RequestMapping用来定义访问的URL，你可以为整个类定义一个

@RequestMapping，或者为每个方法指定一个。

把@RequestMapping放在类级别上，这可令它与方法级别上的

@RequestMapping注解协同工作，取得缩小选择范围的效果。

例如：

@RequestMapping("/test")

public class TestController {}

则，该类下的所有访问路径都在/test之下。

2.将@RequestMapping用于整个类不是必须的，如果没有配置，所有的方法

的访问路径配置将是完全独立的，没有任何关联。

3.完整的参数项为：@RequestMapping(value="",method =

{"",""},headers={},params={"",""})，各参数说明如下：

value :String[] 设置访问地址

method: RequestMethod[]设置访问方式，字符数组，查看RequestMethod

类，包括GET, HEAD, POST, PUT, DELETE, OPTIONS, TRACE,常用

RequestMethod.GET，RequestMethod.POST

headers:String[] headers一般结合method = RequestMethod.POST使用

params: String[] 访问参数设置，字符数组 例如：userId=id

4.value的配置还可以采用模版变量的形式 ，例如：@RequestMapping

(value="/owners/{ownerId}", method=RequestMethod.GET)，这点将在介

绍@PathVariable中详细说明。

5.@RequestMapping params的补充说明，你可以通过设置参数条件来限制

访问地址，例如params="myParam=myValue"表达式，访问地址中参数只有

包含了该规定的值"myParam=myValue"才能匹配得上，类似"myParam"之类

的表达式也是支持的，表示当前请求的地址必须有该参数(参数的值可以是

任意)，"!myParam"之类的表达式表明当前请求的地址不能包含具体指定的

参数"myParam"。

6.有一点需要注意的，如果为类定义了访问地址为\*.do,\*.html之类的，则

在方法级的@RequestMapping，不能再定义value值，否则会报错，例如

Java代码

@RequestMapping("/bbs.do")

public class BbsController {

@RequestMapping(params = "method=getList")

public String getList() {

return "list";

}

@RequestMapping(value= "/spList")

public String getSpecialList() {

return "splist";

}

}

如上例：/bbs.do?method=getList 可以访问到方法getList() ；而访

问/bbs.do/spList则会报错.

@PathVariable

1.@PathVariable用于方法中的参数，表示方法参数绑定到地址URL的模板

变量。

例如：

Java代码

@RequestMapping(value="/owners/{ownerId}",

method=RequestMethod.GET)

public String findOwner(@PathVariable String ownerId, Model

model) {

Owner owner = ownerService.findOwner(ownerId);

model.addAttribute("owner", owner);

return "displayOwner";

}

2.@PathVariable用于地址栏使用{xxx}模版变量时使用。

如果@RequestMapping没有定义类似"/{ownerId}" ，这种变量，则使用在

方法中@PathVariable会报错。

@ModelAttribute

1.应用于方法参数,参数可以在页面直接获取，相当于

request.setAttribute(,)

2.应用于方法,将任何一个拥有返回值的方法标注上 @ModelAttribute，使

其返回值将会进入到模型对象的属性列表中.

3.应用于方法参数时@ModelAttribute("xx"),须关联到Object的数据类型

，基本数据类型 如：int,String不起作用

例如：

Java代码

@ModelAttribute("items")//<——①向模型对象中添加一个名为items的

属性

public List<String> populateItems() {

List<String> lists = new ArrayList<String>();

lists.add("item1");

lists.add("item2");

return lists;

}

@RequestMapping(params = "method=listAllBoard")

public String listAllBoard(@ModelAttribute("currUser")User user,

ModelMap model) {

bbtForumService.getAllBoard();

//<——②在此访问模型中的items属性

System.out.println("model.items:" + ((List<String>)

model.get("items")).size());

return "listBoard";

}

在 ① 处，通过使用 @ModelAttribute 注解，populateItem() 方法将在

任何请求处理方法执行前调用，Spring MVC 会将该方法返回值以“items

”为名放入到隐含的模型对象属性列表中。

所以在 ② 处，我们就可以通过 ModelMap 入参访问到 items 属性，当执

行 listAllBoard() 请求处理方法时，② 处将在控制台打印

出“model.items:2”的信息。当然我们也可以在请求的视图中访问到模型

对象中的 items 属性。

@ResponseBody

这个注解可以直接放在方法上，表示返回类型将会直接作为HTTP响应字节

流输出(不被放置在Model，也不被拦截为视图页面名称)。可以用于ajax。

@RequestParam

@RequestParam是一个可选参数，例如：@RequestParam("id") 注解，所以

它将和URL所带参数 id进行绑定

如果入参是基本数据类型（如 int、long、float 等），URL 请求参数中

一定要有对应的参数，否则将抛出

org.springframework.web.util.NestedServletException 异常，提示无

法将 null 转换为基本数据类型.

@RequestParam包含3个配置 @RequestParam(required = ,value="",

defaultValue = "")

required :参数是否必须，boolean类型,可选项，默认为true

value: 传递的参数名称，String类型,可选项，如果有值，对应到设置方

法的参数

defaultValue:String类型,参数没有传递时为参数默认指定的值

@SessionAttributes session管理

Spring 允许我们有选择地指定 ModelMap 中的哪些属性需要转存到

session 中，以便下一个请求属对应的 ModelMap 的属性列表中还能访问

到这些属性。这一功能是通过类定义处标注 @SessionAttributes 注解来

实现的。@SessionAttributes 只能声明在类上，而不能声明在方法上。

例如

@SessionAttributes("currUser") // 将ModelMap 中属性名为currUser 的属性

@SessionAttributes({"attr1","attr2"})

@SessionAttributes(types = User.class)

@SessionAttributes(types = {User.class,Dept.class})

@SessionAttributes(types = {User.class,Dept.class},value={"attr1","attr2"})

@CookieValue 获取cookie信息

@RequestHeader 获取请求的头部信息