**【1】springmvc常用注解之@Controller和@RequestMapping**

发表于7个月前(2014-09-20 19:15) 阅读（3700） | 评论（[1](http://my.oschina.net/zhdkn/blog/316530" \l "comments)） *10*人收藏此文章, [我要收藏](javascript:add_to_favor(316530,3))

*赞1*

[**慕课网，程序员升职加薪神器，点击免费学习**](http://www.imooc.com/course/list?from=oschina)

*摘要*springmvc常用注解之@Controller和@RequestMapping，详细说明@RequestMapping的常用属性

[springmvc](http://www.oschina.net/search?scope=blog&q=springmvc)[@Controller](http://www.oschina.net/search?scope=blog&q=%40Controller)[@RequestMapping](http://www.oschina.net/search?scope=blog&q=%40RequestMapping)[注解](http://www.oschina.net/search?scope=blog&q=%E6%B3%A8%E8%A7%A3)

对于各种注解而言，排第一的当然是“@Controller”,表明某类是一个controller。

“@RequestMapping”请求路径映射，如果标注在某个controller的类级别上，则表明访问此类路径下的方法都要加上其配置的路径；最常用是标注在方法上，表明哪个具体的方法来接受处理某次请求。

[?](http://my.oschina.net/zhdkn/blog/316530)

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | @Controller  @RequestMapping(value="/book")  public class BookController {  @RequestMapping(value="/title")  public String getTitle(){  return "title";  }  @RequestMapping(value="/content")  public String getContent(){  return "content";  }  } |

由于BookController类加了value="/book"的“@RequestMapping”的注解，所有相关路径都要加上"/book"，即请求的url分别为：

1. http://localhost:8080/book/title
2. http://localhost:8080/book/content

"@RequestMapping"的value值前后是否有“/”对请求的路径没有影响，即value="book" 、"/book"、"/book/"其效果是一样的

"@RequestMapping" 的属性

* value： 指定请求的实际url  
  1. 普通的具体值。如前面的value="/book"。  
  2. 含某变量的一类值

[?](http://my.oschina.net/zhdkn/blog/316530)

|  |  |
| --- | --- |
| 1  2  3  4  5 | @RequestMapping(value="/get/{bookId}")  public String getBookById(@PathVariable String bookId,Model model){  model.addAttribute("bookId", bookId);  return "book";  } |

路径中的bookId可以当变量，@PathVariable 注解即提取路径中的变量值

3.ant风格  
@RequestMapping(value="/get/id?")：可匹配“/get/id1”或“/get/ida”，但不匹配“/get/id”或“/get/idaa”;  
@RequestMapping(value="/get/id\*")：可匹配“/get/idabc”或“/get/id”，但不匹配“/get/idabc/abc”;  
@RequestMapping(value="/get/id/\*")：可匹配“/get/id/abc”，但不匹配“/get/idabc”;  
@RequestMapping(value="/get/id/\*\*/{id}")：可匹配“/get/id/abc/abc/123”或“/get/id/123”，也就是Ant风格和URI模板变量风格可混用;  
  
4. 含正则表达式的一类值  
@RequestMapping(value="/get/{idPre:\\d+}-{idNum:\\d+}")：可以匹配“/get/123-1”，但不能匹配“/get/abc-1”，这样可以设计更加严格的规则。  
可以通过@PathVariable 注解提取路径中的变量(idPre,idNum)  
  
5. 或关系  
@RequestMapping(value={"/get","/fetch"} )即 /get或/fetch都会映射到该方法上

* method： 指定请求的method类型， GET、POST、PUT、DELETE等；  
  @RequestMapping(value="/get/{bookid}",method={RequestMethod.GET,RequestMethod.POST})
* params  
  @RequestMapping(params="action=del")，请求参数包含“action=del”,如：http://localhost:8080/book?action=del
* headers  
  @RequestMapping(value="/header/id", headers = "Accept=application/json")：表示请求的URL必须为“/header/id 且 请求头中必须有“Accept =application/json”参数即可匹配。

### **【2】[Spring mvc中@RequestMapping 6个基本用法小结](http://jackyrong.iteye.com/blog/1806326)**

**博客分类：**

* [JAVA相关](http://jackyrong.iteye.com/category/40691)

[spring](http://www.iteye.com/blogs/tag/spring)[mvc](http://www.iteye.com/blogs/tag/mvc)

小结下spring mvc中的@RequestMapping的用法。  
  
1）最基本的，方法级别上应用，例如：

Java代码 [复制代码](http://jackyrong.iteye.com/blog/1806326)[[收藏代码http://jackyrong.iteye.com/images/spinner.gif](javascript:void())](javascript:void())

1. @RequestMapping(value="/departments")
2. public String simplePattern(){
3. System.out.println("simplePattern method was called");
4. return "someResult";
5. }

@RequestMapping(value="/departments")

public String simplePattern(){

System.out.println("simplePattern method was called");

return "someResult";

}

则访问http://localhost/xxxx/departments的时候，会调用 simplePattern方法了  
  
2） 参数绑定

Java代码 [复制代码](http://jackyrong.iteye.com/blog/1806326)[[收藏代码http://jackyrong.iteye.com/images/spinner.gif](javascript:void())](javascript:void())

1. @RequestMapping(value="/departments")
2. public String findDepatment(
3. @RequestParam("departmentId") String departmentId){
4. System.out.println("Find department with ID: " + departmentId);
5. return "someResult";
6. }

@RequestMapping(value="/departments")

public String findDepatment(

@RequestParam("departmentId") String departmentId){

System.out.println("Find department with ID: " + departmentId);

return "someResult";

}

形如这样的访问形式：  
  
/departments?departmentId=23就可以触发访问findDepatment方法了  
  
3 REST风格的参数

Java代码 [复制代码](http://jackyrong.iteye.com/blog/1806326)[[收藏代码http://jackyrong.iteye.com/images/spinner.gif](javascript:void())](javascript:void())

1. @RequestMapping(value="/departments/{departmentId}")
2. public String findDepatment(@PathVariable String departmentId){
3. System.out.println("Find department with ID: " + departmentId);
4. return "someResult";
5. }

@RequestMapping(value="/departments/{departmentId}")

public String findDepatment(@PathVariable String departmentId){

System.out.println("Find department with ID: " + departmentId);

return "someResult";

}

形如REST风格的地址访问，比如：  
/departments/23，其中用(@PathVariable接收rest风格的参数  
  
4 REST风格的参数绑定形式之2  
先看例子，这个有点象之前的：

Java代码 [复制代码](http://jackyrong.iteye.com/blog/1806326)[[收藏代码http://jackyrong.iteye.com/images/spinner.gif](javascript:void())](javascript:void())

1. @RequestMapping(value="/departments/{departmentId}")
2. public String findDepatmentAlternative(
3. @PathVariable("departmentId") String someDepartmentId){
4. System.out.println("Find department with ID: " + someDepartmentId);
5. return "someResult";
6. }

@RequestMapping(value="/departments/{departmentId}")

public String findDepatmentAlternative(

@PathVariable("departmentId") String someDepartmentId){

System.out.println("Find department with ID: " + someDepartmentId);

return "someResult";

}

这个有点不同，就是接收形如/departments/23的URL访问，把23作为传入的departmetnId,，但是在实际的方法findDepatmentAlternative中，使用  
@PathVariable("departmentId") String someDepartmentId，将其绑定为  
someDepartmentId,所以这里someDepartmentId为23  
  
5 url中同时绑定多个id

Java代码 [复制代码](http://jackyrong.iteye.com/blog/1806326)[[收藏代码http://jackyrong.iteye.com/images/spinner.gif](javascript:void())](javascript:void())

1. @RequestMapping(value="/departments/{departmentId}/employees/{employeeId}")
2. public String findEmployee(
3. @PathVariable String departmentId,
4. @PathVariable String employeeId){
5. System.out.println("Find employee with ID: " + employeeId +
6. " from department: " + departmentId);
7. return "someResult";
8. }

@RequestMapping(value="/departments/{departmentId}/employees/{employeeId}")

public String findEmployee(

@PathVariable String departmentId,

@PathVariable String employeeId){

System.out.println("Find employee with ID: " + employeeId +

" from department: " + departmentId);

return "someResult";

}

这个其实也比较好理解了。  
  
6 支持正则表达式

Java代码 [复制代码](http://jackyrong.iteye.com/blog/1806326)[[收藏代码http://jackyrong.iteye.com/images/spinner.gif](javascript:void())](javascript:void())

1. @RequestMapping(value="/{textualPart:[a-z-]+}.{numericPart:[\\d]+}")
2. public String regularExpression(
3. @PathVariable String textualPart,
4. @PathVariable String numericPart){
5. System.out.println("Textual part: " + textualPart +
6. ", numeric part: " + numericPart);
7. return "someResult";
8. }

@RequestMapping(value="/{textualPart:[a-z-]+}.{numericPart:[\\d]+}")

public String regularExpression(

@PathVariable String textualPart,

@PathVariable String numericPart){

System.out.println("Textual part: " + textualPart +

", numeric part: " + numericPart);

return "someResult";

}

比如如下的URL：/sometext.123，则输出：  
Textual part: sometext, numeric part: 123.