SpringMVC的请求地址映射

# 相关注解及类

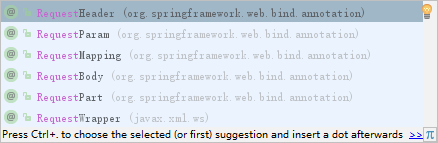
## 相关注解

**@RequestMapping(最重要)、@RequestParam、@PathVariable**

注意：@RequestParam与@PathVariable的区别。

**@RequestHeader、@RequestBody**、@RequestPart、@RequestWrapper

**@CookieValue**



## 用到的类

枚举类：RequestMethod

## 请求地址映射

### 精准化映射

利用RequestMapping的**value、method、params、header**实现；同时利用@RequestParam对请求参数添加限制条件。

### @PathVariable

从请求地址URL模板中提取变量值。

### Ant风格匹配：模糊匹配。

### 正则表达式匹配

### “或”匹配：从多个条件中满足一个即可。

# @RequestMapping：映射请求地址。

## @RequestMapping功能：映射请求地址

Spring MVC使用**@RequestMapping注解**为控制器**类或内部方法**指定可以处理哪些URL请求。**@RequestMapping**是一个用来**处理请求地址映射**的注解（将请求地址映射到对应的控制器方法中）**，**可用于**类或方法**上。

@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。

## @RequestMapping修饰类和方法

注解**@RequestMapping**在**类级别和方法级别**层面都可标注，用来确定将被调用方法的URL。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**；用于方法上，表示**在类的父路径下追加方法上注解中的地址**将会访问到该方法。在类上可以使用也可以不使用@RequestMapping，但是在handle方法上必须使用**@RequestMapping**。

按照一般的开发习惯：类上的@RequestMapping一般就是利用**类名**作为**父路径，可以指定多层**，如@RequestMapping(value='/xxx/xxx/类名')；然后内部方法上利用**方法名**指定，一般就是一层路径，如@RequestMapping(value='/方法名')。

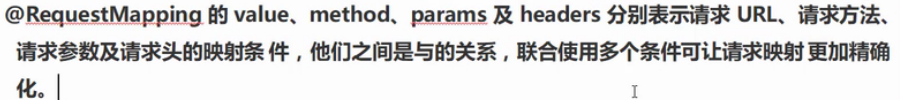
提供**初步的请求映射信息**，相当于WEB应用的根目录。

在标有**@Controller**的类中，利用**@RequestMapping**可以将该类中的方法变成**handler**方法，用来处理请求。



## @RequestMapping的属性（6个）：实现精准化映射

**@RequestMapping具有四个属性：value、method、params、headers、consumes、produces**。(consumes、produces使用较少)



分别为映射请求路径（**value属性）**、请求方法（**method属性**）、请求参数（**parmas属性**）或请求头（**headers属性**）。多个属性同时指定的时候，利用逗号隔开，属性之间的关系是“与”的关系。

**@RequestMapping**一般情况必须指定属性value，用来指定对应的URL，这就是粗略映射。当@RequestMapping**只指定一个属性value时**，此时value可省略不写，即@RequestMapping(value="/hello")等价于@RequestMapping("/hello")。

注意："@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。

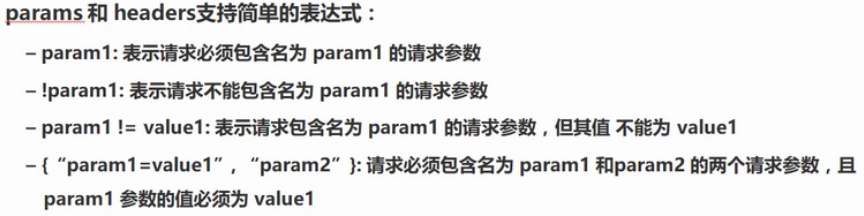
### method：指定请求的method类型， GET、POST、PUT、DELETE等；

@RequestMapping(value="/get/{bookid}",method={RequestMethod.GET,RequestMethod.POST})

RequestMethod是一个枚举类。

### params：指定request中必须包含某些参数值是，才让该方法处理。

@RequestMapping(params="action=del")，请求参数包含“action=del”,如：<http://localhost:8080/book?action=del>

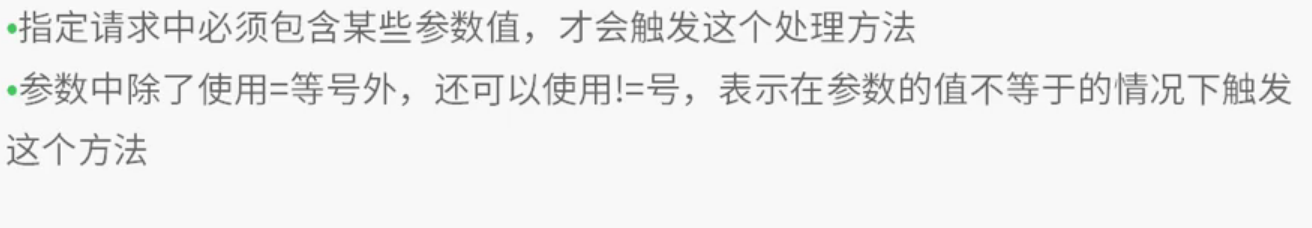


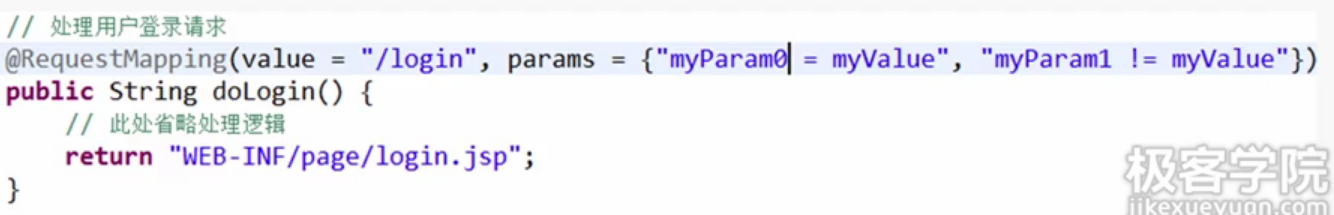
**params**通过一个数组定义,如**params={"xxx1=yyy1","xxx2=yyy2","xxx3","xxx4"}**

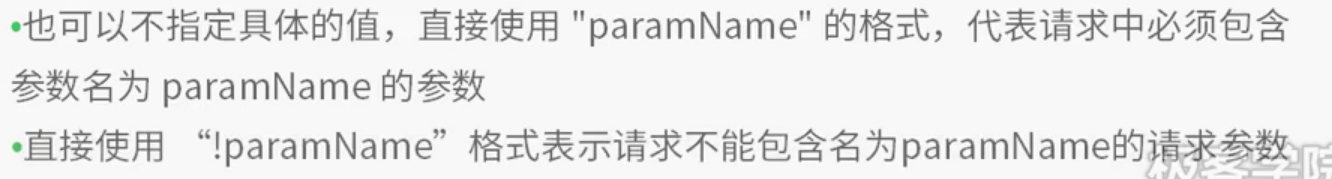
意思是，必须有参数值为yyy1的xxx1参数，值为yyy2的xxx2参数，必须具有参数xxx3和xxx4参数，值可以为任意。

如：@RequestMapping(value="/hello",method=RequestMethod.POST,params={"user=zhaohong","passwd=zhao12","code"})









### 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。

**headers**的使用方法与params相同，如headers={"Accept=text/html"}

@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。



### consumes：指定处理请求的提交内容类型（Content-Type），例如application/json, text/html。

@Controller

@RequestMapping(value = "/pets", method = RequestMethod.POST, consumes="application/json")

public void addPet(@RequestBody Pet pet, Model model) {

// implementation omitted

}

方法仅处理request Content-Type为“application/json”类型的请求。





### 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。

@Controller

@RequestMapping(value = "/pets/{petId}", method = RequestMethod.GET, **produces="application/json")**

@ResponseBody

public Pet getPet(@PathVariable String petId, Model model) {

// implementation omitted

}

方法仅处理request请求中Accept头中包含了"application/json"的请求，同时暗示了返回的内容类型为application/json;

## 或的关系：用花括号括起来，逗号隔开即可。

@RequestMapping(**value={"/getBody","/fetchBody"}** )即 /get或/fetch都会映射到该方法上。

@RequestMapping(value=**"getBody"**,**method={RequestMethod.*GET*,RequestMethod.*POST*}**)

# Ant风格：URL支持的3种风格

## Ant风格资源地址支持3种匹配符介绍

### ? : 匹配文件名中的一个字符；

### \*：匹配文件名中的任意多个字符(包括0)，但是只能匹配1层目录；

### \*\*:匹配任意多层目录(包括0)，可以为0层、1层或多层目录；

## 示例

@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**。



# @PathVariable：用来获取请求路径URL中的动态参数

## REST风格的URL

**带占位符的 URL** 是 Spring3.0 新增的功能,该功能在SpringMVC 向 **REST 目标**挺进发展过程中具有里程碑的意义。

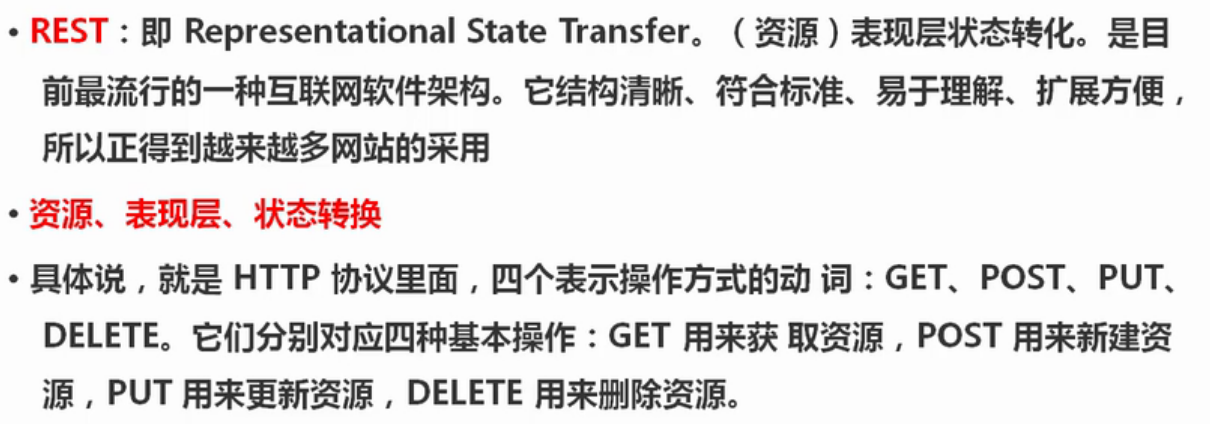
REST风格简单描述：

传统URL：[http://www.baidu.com?**user=zhaohong&password=zhao12**](http://www.baidu.com?user=zhaohong&password=zhao12)

REST风格：[http://www.baidu.com/**zhaohong/zhao12**](http://www.baidu.com/zhaohong/zhao12)

就是改变**key=value**的形式，直接使用value。

## REST：表现层状态转化



POST：新建资源；（**增**）

DELETE：删除资源；（**删**）

PUT：更新资源；（**改**）

GET:获取资源。（**查**）

## 映射信息：占位符

SpringMVC配置映射信息，需要在映射信息后面对应的位置加入**占位符{xxx}**。

如：@RequestMapping(value="www/baidu/com/{user}/{passwd}"}

## @PathVariable的功能

@PathVariable用于将**请求URL中的模板变量值**映射到功能**处理方法的参数**上，即提取URL模板中的变量作为参数。。

## @PathVariable的value属性

通过 **@PathVariable** 可以将 **URL 中占位符参数**绑定到**控制器处理方法的入参**中:URL 中的 **{xxx} 占位符**可以通过**@PathVariable(value="xxx")** 绑定到**操作方法的入参（形参）**中。如果URL模板中定义的{XXX}变量名与方法参数名一致，value属性可以省略，若不一致，必须制定value属性，但是如果只定义value一个属性，**value**往往也可以省略。

## 正则匹配URL与@PathVariable的结合

@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)

@ResponseBody  
@RequestMapping(value = **"getRegexParam/{idPre:\\d+}-{idNum:\\d+}"**,method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getRegexParams(@PathVariable(**"idPre"**) **int** idPre,@PathVariable String idNum){  
 **return** idPre+**"-"**+idNum;  
}

<http://localhost:8080/SpringMVCDemo0711/test/getRegexParam/666-888>

结果：666-888

## 示例：

@ResponseBody  
@RequestMapping(value = **"/getPathVairable/{id}/{name}/{address}",**method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getPathVariableValue**(@PathVariable(value = "id") String NO,@PathVariable("name")String myName,@PathVariable String address**){  
 **return "StudentNo:"**+NO+**"<br>StudentName:"**+myName+**"<br>StudentAddress:"**+address;  
}

这个例子中：{id}与方法参数NO对应；{name}与方法参数myName对应；{address}与方法参数address对应。

# @RequestParam绑定单个请求参数值

## @RequestParam的作用

**@RequestParam**用于将**请求参数区数据**映射到**功能处理方法的参数**上。

public String requestparam1(@RequestParam String username)

请求中包含username参数（如/requestparam1?username=zhang），则自动传入。

注意：在Controller的handler方法中获取参数的方式两种:一种是**request.getParameter("name")**，另外一种是用注解**@RequestParam**直接获取。

显然直接使用@RequestParam比较方便。

当然，现在不添加@RequestParam，也已经默认支持

@ResponseBody  
@RequestMapping(value = **"/getParams"**,method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getParams(String id,String name){  
 **return "StudentID:"**+id+**"<br>StudentName:"**+name;  
}

http://localhost:8080/SpringMVCDemo0711/test/getParams?id=100&name=zhaohong

StudentID:100

StudentName:zhaohong

**注意:**



## @RequestParam的3个属性：

**value**：参数名字，即入参的请求参数名字，如username表示请求的参数区中的名字为username的参数的值将传入；

**required**：是否必须，**默认是true**，表示请求中一定要有相应的参数，否则将抛出异常；

**defaultValue**：默认值，表示如果请求中没有同名参数时的默认值，设置该参数时，自动将required设为false。

## 示例

public String requestparam4(@RequestParam(value="username",required=false) String username)

表示请求中可以没有名字为username的参数，如果没有默认为null。

## 需要注意如下几点：

**原子类型**：必须有值，否则抛出异常，如果允许空值请使用包装类代替。

**Boolean包装类型**：默认Boolean.FALSE，其他引用类型默认为null。

## POJO对象接收：根据对象属性自动创建对象

当然需要Pojo中有各个变量的set方法。

@ResponseBody  
@RequestMapping(value = **"/getStudent"**,method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getStudent(Student student){  
 **return** student.toString();  
}

http://localhost:8080/SpringMVCDemo0711/test/getStudent?id=100&name=zhaohong&address=Beijing

结果：Student{id=100, name='zhaohong', address='Beijing'}

## 多个同名参数接收问题：集合接收或数组接收

如果请求中有多个同名的应该如何接收呢？

方法：利用**数组或者集合**接收。

举例：给用户角色分配时，可能授予多个角色权限：

### 利用List集合接收

@ResponseBody  
@RequestMapping(value = **"getList"**,method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getList(@RequestParam(value = **"role"**)List<String> roleList){  
 **return** roleList.toString();  
}

<http://localhost:8080/SpringMVCDemo0711/test/getList?role=student&role=teacher&role=worker>

结果：[student, teacher, worker]

### 利用数组接收

@ResponseBody  
@RequestMapping(value = **"getArray"**,method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getArray(@RequestParam(value = **"role"**)String[] roleArray){  
 **return** Arrays.*toString*(roleArray);  
}

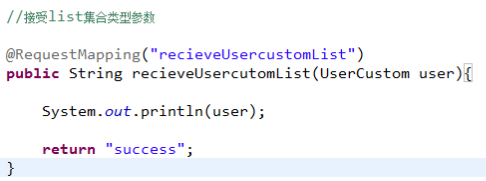
<http://localhost:8080/SpringMVCDemo0711/test/getArray?role=student&role=teacher&role=worker>

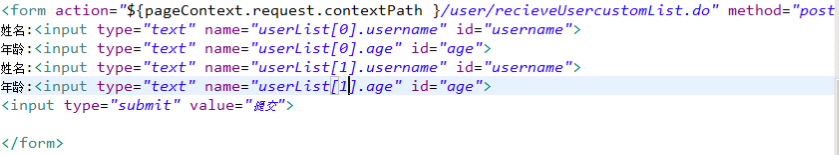
结果：[student, teacher, worker]

## 从页面接受List和Map参数

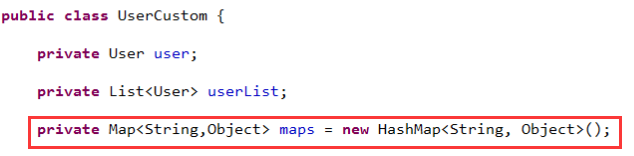
### 接收list集合类型参数



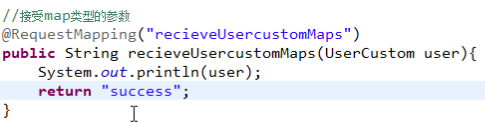


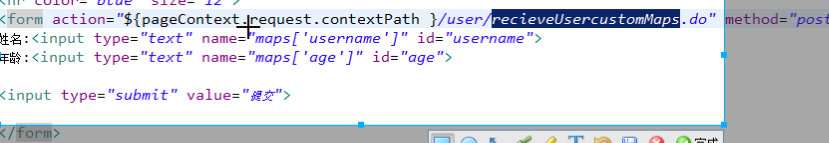


### 接收map类型参数：



**建议对map进行初始化一下。**





# @PathVariable与@RequestParam的区别

## 相同点

### 作用位置相同：都是直接修饰方法参数变量；

### 功能相似：都是将URL中的变量值映射到方法参数上；

### 都具有value属性：将URL变量名与方法参数名映射起来；

## 不同点

### 对应的URL不同

@PathVariable是从请求路径中提取变量值，对应的URL是REST风格，具有占位符{XXX}，即URL模板；如/{id}/{name}

@RequestParam是从请求参数中获取变量值，对应的URL是传统URL，key=value形式，如?id=1&name=zhaohong；其是从参数中获取value，get请求是添加在路径后面，对于post请求的参数也可以获取。

### 设置默认值

@RequestParam可以通过**defaultValue属性**设置默认值，而@PathVariable不可以。

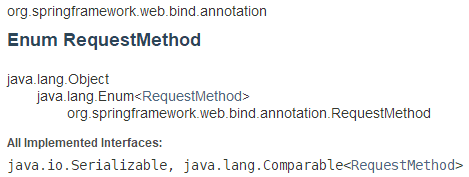
### 是否必需

@RequestParam可以通过**required属性**设置是否必需，默认为true；而@PathVariable一定是必需的。

# RequestMethod枚举类的介绍

## RequestMethod类：定义HTTP的请求方法。

public enum **RequestMethod** extends java.lang.Enum<RequestMethod>



**Java 5 enumeration of HTTP request methods**. Intended for use with the **RequestMapping.method()** attribute of the RequestMapping annotation.

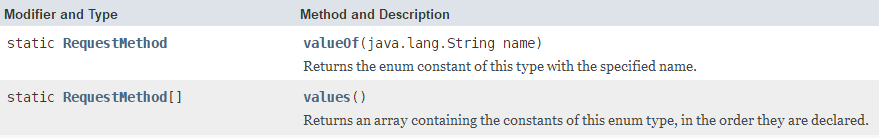
DispatcherServlet默认不支持trace和options请求。

Note that, **by default**, [**DispatcherServlet**](https://docs.spring.io/spring/docs/5.0.5.BUILD-SNAPSHOT/javadoc-api/org/springframework/web/servlet/DispatcherServlet.html)**supports GET, HEAD, POST, PUT, PATCH and DELETE only**. DispatcherServlet will process TRACE and OPTIONS with the default HttpServlet behavior **unless explicitly told to dispatch those request types** as well: Check out the "dispatchOptionsRequest" and "dispatchTraceRequest" properties, switching them to "true" if necessary.

## 枚举元素（8个）Enum Constants

**DELETE 、GET 、HEAD、PATCH 、POST 、PUT** 、TRACE、OPTIONS

## 2个静态方法



# @RequestHeader注解

## 与@RequestParam类似

@RequestParam是从request的param中获取值并绑定到方法的参数上；

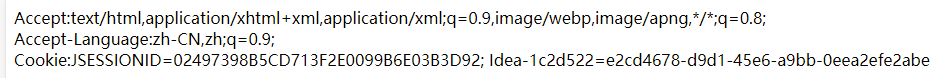
@RequestHeader是从request的header中获取值并绑定到方法的参数上；

## 示例：@RequestHeader示例：

@ResponseBody  
@RequestMapping(value = **"getHeader"**,method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getHeader**(@RequestHeader("Accept") String accept,@RequestHeader("Accept-Language") String language,  
 @RequestHeader("Cookie") String cookie**){  
 **return "Accept:"**+accept+**";\nAccept-Language:"**+language+**";\nCookie:"**+cookie;

}

结果：

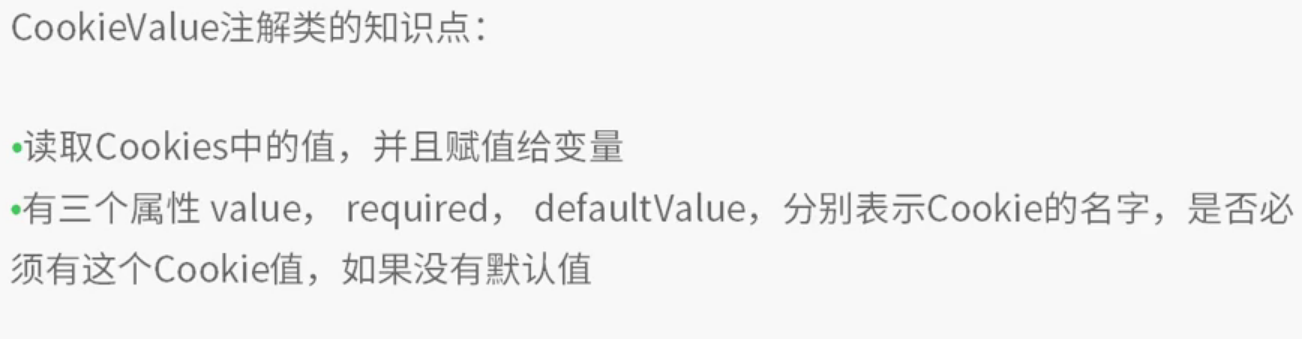


# @RequestBody注解(使用较少)

见<*@ResponseBody与@RequestBody>注解*介绍。

# @CookieValue(使用较少)

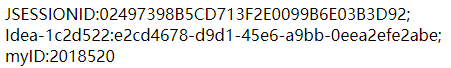
@CookieValue:从Cookie中获取变量值。@CookieValue与@RequestParam类似，都是3个属性：**value、required、defaultValue**。



示例：

@ResponseBody  
@RequestMapping(value = **"getCookieValue"**,method = {RequestMethod.***GET***,RequestMethod.***POST***})  
**public** String getCookieValue(@CookieValue(**"JSESSIONID"**) String sessionID,@CookieValue(**"Idea-1c2d522"**) String idea, @CookieValue(value = **"myId"**,defaultValue = **"2018520"**,required = **false**) String myID){  
 **return "JSESSIONID:"**+sessionID+**";<br>Idea-1c2d522:"**+idea+**";<br>myID:"**+myID;  
}

结果：



# @SessionAttributes

