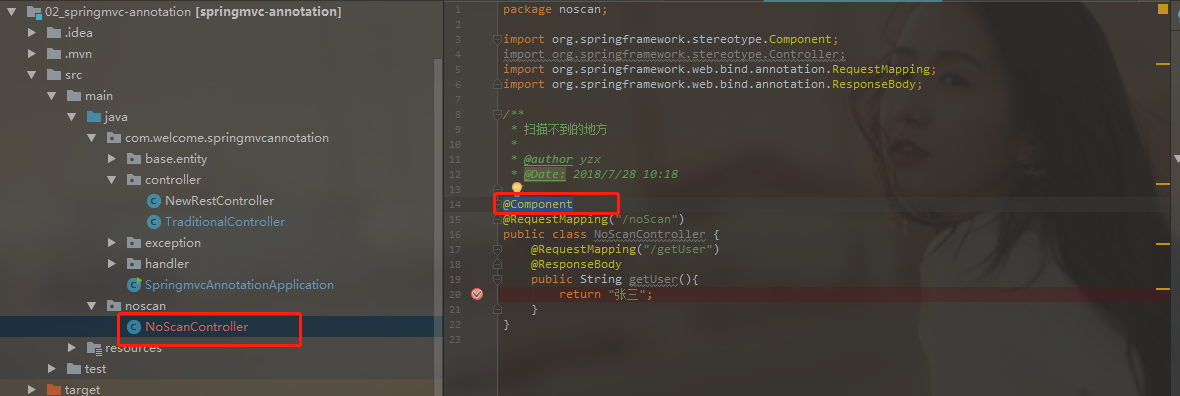
**Spring MVC 注解**

**前言：Spring 2.5开始引入注解，随着Spring 版本的更新，注解也在不断的扩展，让开发者的工作变得轻松。**

## 1. 组件类注解（使用在类上，纳入进spring容器中管理）

@Repository、@Service、@Controller 和 @Component

其实前三个组件和@Component都是等效的，只是根据名字很容易区分出来层次，所以如果明确类所在的层次最好用对应的注解，这样更加清晰



看，就算我在controller中用的事@Component注解依然可以正常访问

### @Controller（控制层组件）

使用它标记的类就是一个SpringMVC Controller 对象，就是一个控制器类，Spring 通过扫描机制查找程序中所有的控制器类，分发处理器会扫描方法上是否使用了@RequestMapping注解，使用了@RequestMapping注解的方法才是真正的处理请求的处理器。

注意\*，只配有@Controller 注解的控制器，返回的其实是页面，如果返回的是String，则表示页面的名字。看代码例子



项目上不推荐这么用，页面跳转统一在前端处理

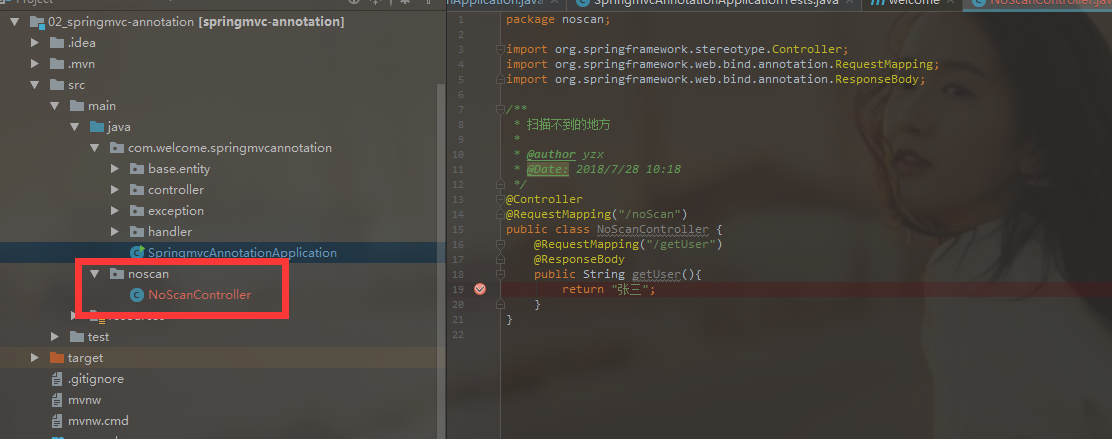
如果想返回具体的数据配合@ResponseBody使用

**超纲：**

传统方式：spring配置文件中

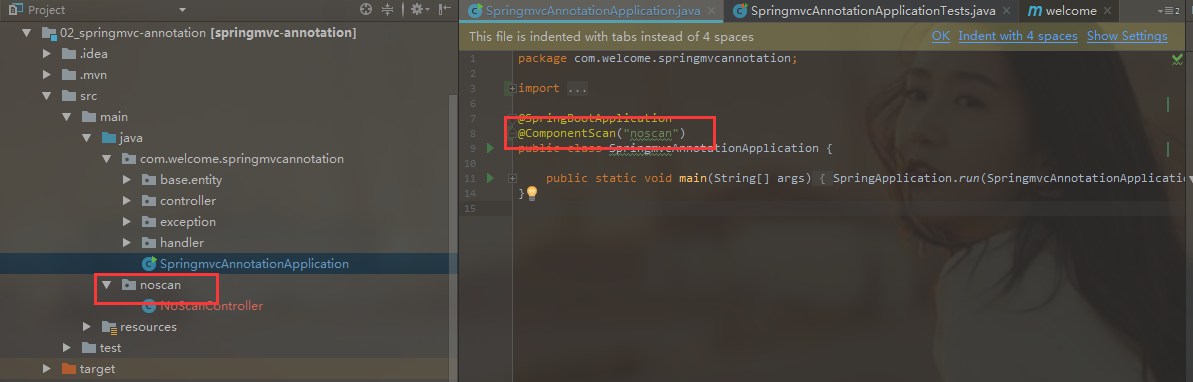
<context:component-scanbase-package="com.welcome.annotation">

Spring boot项目的Bean装配默认规则是根据Application类所在的包位置从上往下扫描.就是说只会扫App类同级的及下级子包。用例子演示一下



如果这样的话，这个控制器是不会被扫描到的，如果就是想让它这样如何扫描到呢？

对：通过在启动类上配置@ComponentScan("noscan")



这样就可以扫描到了

### @RestController（控制层组件）

相当于RestController + ResponseBody所以也就很明显了，现在基本都使用它

插播：@ResponseBody

作用：表示该方法的返回的结果直接写入 HTTP 响应正文（ResponseBody）中，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。

**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；

### @Service(业务层组件，加到实现类)

### @repository（数据访问层组件）

### @Component（泛指组件，不好归类的时候可以用这个标识）

## 2请求类注解@RequestMapping

**使用位置：**类或者方法

**作用：**用来映射请求和方法，处理请求动作

作用在类上，表示在本控制器所有请求处理方法外层加了一层路径，也是保证请求唯一性的一个很方便的手段

**属性：**

###### 1. value 注解默认值，如果只有唯一属性，可以省略

将URL映射到方法上

？：有同学可能会问了，我一直都是用的path属性呀，和这个value有啥关系呀

###### 2. method

改属性用来指示该方法仅仅处理哪些HTTP请求方式（种类可以参见：RequestMethod 枚举）

如果没有指定method属性，则请求处理方法可以处理任意的HTTP请求。

Spring4.3中引进了｛@GetMapping、@PostMapping、@PutMapping、@DeleteMapping、@PatchMapping｝

其实就是@RequestMapping 指定了Method 类型，帮助简化常用的HTTP方法的映射，并更好地表达被注解方法的语义。可以查看具体注解源码就能理解

###### 3. consumes（实现请求过滤）

该属性指定处理请求的提交内容类型（Content-Type）

常用类型：

application/json（json数据格式）

application/x-www-form-urlencoded （表单提交格式）

multipart/form-data （附件上传格式）

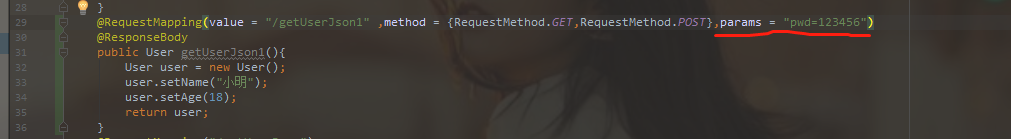
…感兴趣的可以自己进行扩展

###### 4. produces (实现请求过滤)

该属性指定返回的内容类型，返回的内容类型必须是request请求头（Accept）中包含的类型。

###### 5. params(实现请求过滤)

该属性指定request中必须包含某些参数值时，才让该方法处理



\*pwd做分析

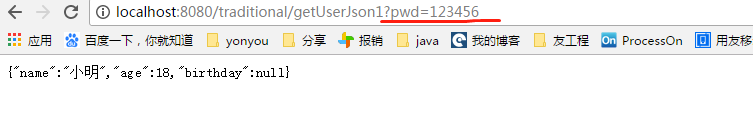
pwd: 表示请求必须包含名为pwd的请求参数

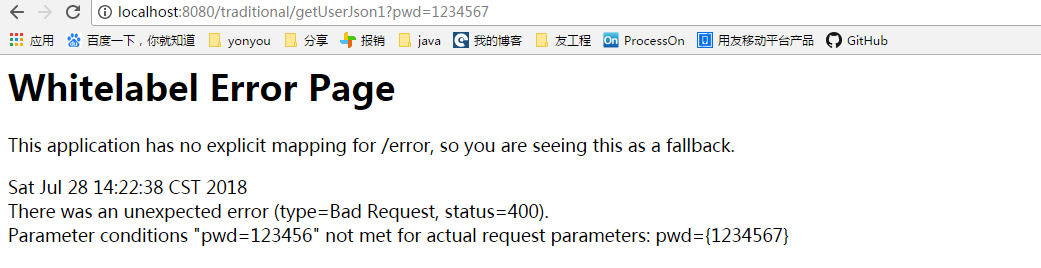
!pwd: 表示请求不能包含名为pwd的请求参数

pwd!= 123456: 表示请求包含名为pwd的请求参数，但其值不能为123456

{“pwd=123456”, “name”}: 请求必须包含名为pwd和name的两个请求参数，且pwd参数的值必须为123456

正确请求

错误请求



###### 6. headers(实现请求过滤)

该属性指定request中必须包含指定的header值，才能让该方法处理请求。

例如：@RequestMapping(value = "/headers", headers = {"Host=localhost:8080"})

Request-header中需要包含Host=localhost:8080让该方法处理请求

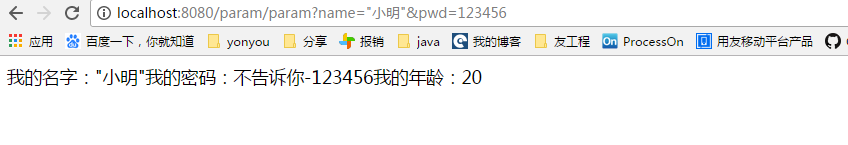
## 3. 参数绑定注解

## 3.1 @RequestParma 注解（位置：方法参数中）

**作用：**该注解用于将指定的请求参数赋值给方法中的形参。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **是否必须** | **解释** |
| **name** | **String** | **否** | **请求参数名称** |
| **value** | **String** | **否** | **和name互为别名** |
| **required** | **Boolean（默认：true）** | **否** | **参数是否为必须** |
| **defaultValue** | **String** | **否** | **如果没传参数时使用的默认值** |

@RequestMapping("/param")  
public String param(@RequestParam(name = "name", required = true) String name,  
@RequestParam(name = "pwd") String pwd,  
@RequestParam(name = "age", required = false, defaultValue = "20") String age) {  
return "我的名字："+ name + "我的密码：不告诉你-"+ pwd + "我的年龄："+age;  
}

## 3.2 @PathVariable 注解（位置：方法参数中）

该注解可以很方便地获取请求URL中动态的参数

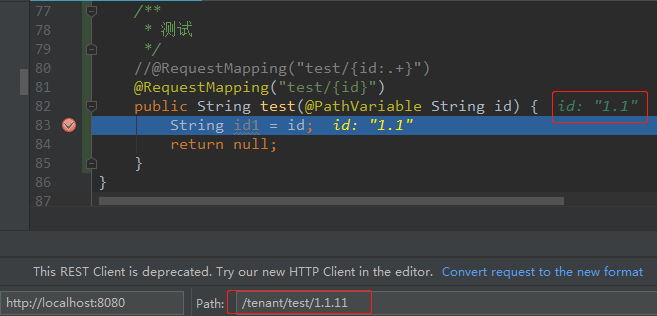
**用法：**RequestMapping 请求中变量：{请求变量名}

方法参数中使用@PathVariable 进行接收此参数，类型要保证和请求地址中类型一致

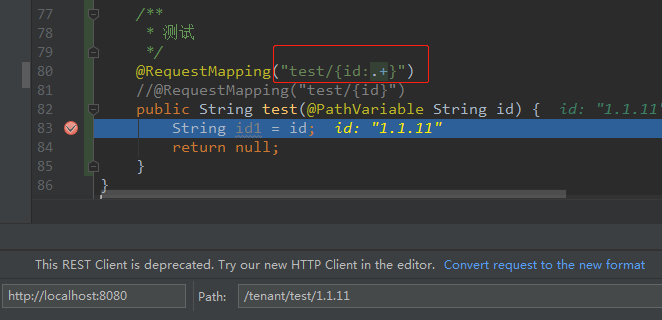
@RequestMapping("/pathVars/{userId}")  
public String pathVars(@PathVariable Integer userId){  
return "我是动态的，我的id:"+ userId;  
}

## 

注:当请求参数存在‘.’时,可能存在以下问题:



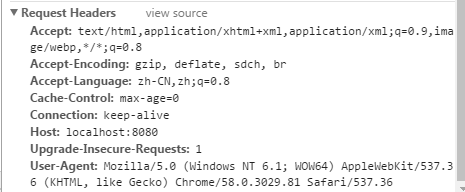
解决方法:

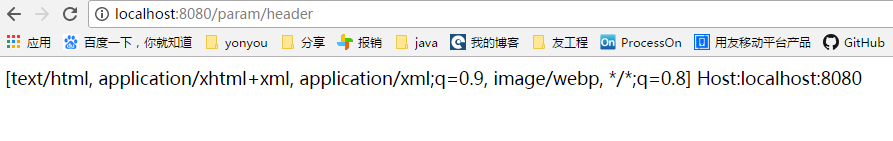


## 3.3 @RequestHeader 注解（位置：方法参数中）

该注解用于请求头信息映射到处理方法的参数上。

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **是否必须** | **解释** |
| **name** | **String** | **否** | **请求参数名称** |
| **value** | **String** | **否** | **和name互为别名** |
| **required** | **Boolean（默认：true）** | **否** | **参数是否为必须** |
| **defaultValue** | **String** | **否** | **如果没传参数时使用的默认值** |



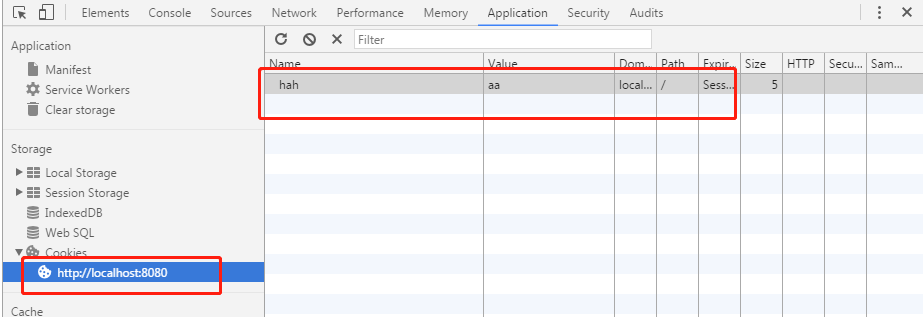
@RequestMapping("/header")  
public String header(@RequestHeader("Accept") String[] accepts, @RequestHeader(value = "Host") String host){  
return Arrays.*toString*(accepts) + " Host:"+ host;  
}

## [3.4 @CookieValue](mailto:3.4@CookieValue)注解（位置：方法参数中）

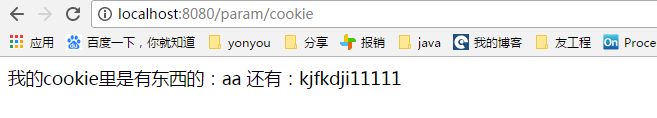
该注解用于将请求的Cookie数据映射到方法参数上。

**属性：**

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **是否必须** | **解释** |
| **name** | **String** | **否** | **请求参数名称** |
| **value** | **String** | **否** | **和name互为别名** |
| **required** | **Boolean（默认：true）** | **否** | **参数是否为必须** |
| **defaultValue** | **String** | **否** | **如果没传参数时使用的默认值** |



@RequestMapping("/cookie")  
public String cookie(@CookieValue("hah") String hahValue,  
@CookieValue(value = "JSESSIONID", required = false, defaultValue = "kjfkdji11111") String sessionId){  
return "我的cookie里是有东西的："+ hahValue +" 还有："+sessionId ;  
}



## 3.5 @SessionAttributes 注解（位置：声明在类上）

该注解可以用来将存储在model中的属性存到HttpSession对象中，

|  |  |  |  |
| --- | --- | --- | --- |
| **属性** | **类型** | **是否必须** | **解释** |
| **name** | **String** | **否** | **请求参数名称** |
| **value** | **String** | **否** | **和name互为别名** |
| **types** | **Class<?>[]** | **否** | **指定欲存的类型** |

@Controller  
@SessionAttributes("user")  
@RequestMapping("/sessionAttrs")  
public class SessionAttrsController {  
@RequestMapping("/getUserPage/{name}")  
public String getUserPage(@PathVariable("name") String name, Model model){  
 User user = new User();  
user.setName(name);  
user.setAge(18);  
model.addAttribute("user",user);  
 return "userPage.html";  
}  
}

页面：

<!DOCTYPE html>  
<html lang="en" xmlns:th="http://www.thymeleaf.org">  
<head>  
<meta charset="UTF-8"/>  
<title>user</title>  
</head>  
<body class="container">  
<div style="margin:auto; width:400px; height:300px;top: 0px;left: 0px;right: 0px;bottom: 0px;position: fixed;">  
<h1>我是个页面</h1>  
  
<div>  
<label for="name">name</label>  
<div class="col-sm-10">  
<h5>model中的值</h5>  
<input type="text" name="name" id="name" th:value="\*{user.name}"/>  
<h5>session中的值</h5>  
<input type="text" name="session" id="session" th:value="\*{session.user.name}"/>  
</div>  
</div>  
</div>  
</body>  
</html>

效果：



**过程是：**当往model中进行存储时，如果类上有@SeesionAttributes注解，并且指定了你存入model中的变量或者类型，那么就会将该值也存入到HttpSession中

## 3.6 @ModelAttribute 注解

该注解将请求参数绑定到model对象上，特别注意，被该注解修饰的方法会在该控制器其他请求方法前执行，如果控制器处理请求的方法是多个的时候需要谨慎使用，感兴趣的可以自己查看一下，这里不做详细讲解

## 格式转换注解@RequestBody

该注解用于读取Request请求中body数据绑定到处理请求方法参数上。

当request header Content-Type:application/json必须通过@RequestBody来处理，也是我们最常用的场景

其他类型的参数可以通过前面讲过的绑定参数的方式获取

@RequestMapping("/getUser")  
public String getUser(@RequestBody User user){  
return "我成功被转了："+ user.getName();  
}

