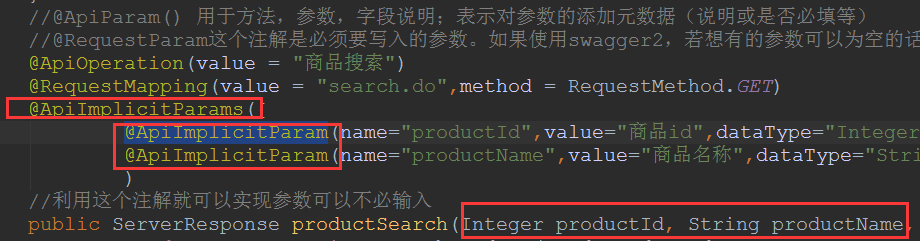
[**swagger2的常用注解，传递参数的注意使用方法**](http://www.cnblogs.com/fengli9998/p/7921601.html)

背景介绍：

刚开始的时候，在controller层使用@RequestParam的时候，发现这个参数是必须要输入值的，但是我们有时候必须查询的时候允许参数为空，使用这个注解就不行了。

在集成了swagger2后，找了半天的原因，发现使用@ApiImplicitParam这个注解可以解决这个问题。



对应下面的参数。

所以我们可以使用这个注解来解决我们所遇到的参考为空的问题。

而且已经集成了swagger2，所以我们尽量来使用这个注解吧。

说明：   
1.这里使用的版本：springfox-swagger2（2.4）springfox-swagger-ui （2.4）   
2.这里是说明常用注解的含义和基本用法（也就是说已经对swagger进行集成完成）   
没有集成的请参见   
[SpringBoot集成springfox-swagger2构建restful API](http://blog.csdn.net/u014231523/article/details/54562695)   
[SpringMVC集成springfox-swagger2构建restful API](http://blog.csdn.net/u014231523/article/details/54411026)   
[官网WIKI](https://github.com/swagger-api/swagger-core/wiki/Annotations-1.5.X#quick-annotation-overview)   
常用注解：   
- **@Api()**用于类；   
表示标识这个类是swagger的资源   
- **@ApiOperation()**用于方法；   
表示一个http请求的操作   
- **@ApiParam()**用于方法，参数，字段说明；   
表示对参数的添加元数据（说明或是否必填等）   
- **@ApiModel()**用于类   
表示对类进行说明，用于参数用实体类接收   
- **@ApiModelProperty()**用于方法，字段   
表示对model属性的说明或者数据操作更改   
- **@ApiIgnore()**用于类，方法，方法参数   
表示这个方法或者类被忽略   
- **@ApiImplicitParam()** 用于方法   
表示单独的请求参数   
- **@ApiImplicitParams()** 用于方法，包含多个 @ApiImplicitParam

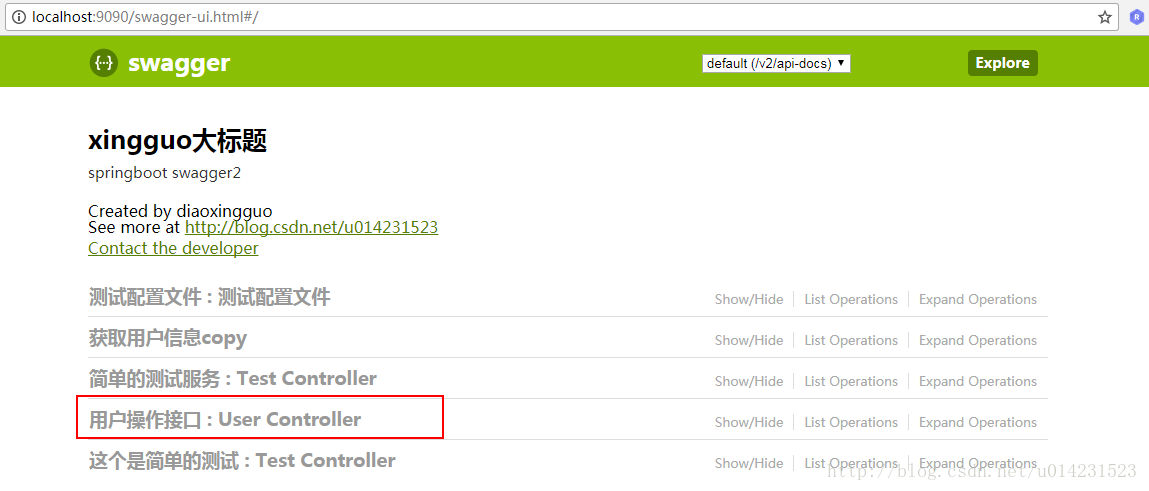
具体使用举例说明：   
**@Api()**   
用于类；表示标识这个类是swagger的资源   
tags–表示说明   
value–也是说明，可以使用tags替代   
**但是tags如果有多个值，会生成多个list**

@Api(value="用户controller",tags={"用户操作接口"})

@RestController

public class UserController {

}

效果图：   


**@ApiOperation()** 用于方法；表示一个http请求的操作   
value用于方法描述   
notes用于提示内容   
tags可以重新分组（视情况而用）   
**@ApiParam()** 用于方法，参数，字段说明；表示对参数的添加元数据（说明或是否必填等）   
name–参数名   
value–参数说明   
required–是否必填

@Api(value="用户controller",tags={"用户操作接口"})

@RestController

public class UserController {

@ApiOperation(value="获取用户信息",tags={"获取用户信息copy"},notes="注意问题点")

@GetMapping("/getUserInfo")

public User getUserInfo(@ApiParam(name="id",value="用户id",required=true) Long id,@ApiParam(name="username",value="用户名") String username) {

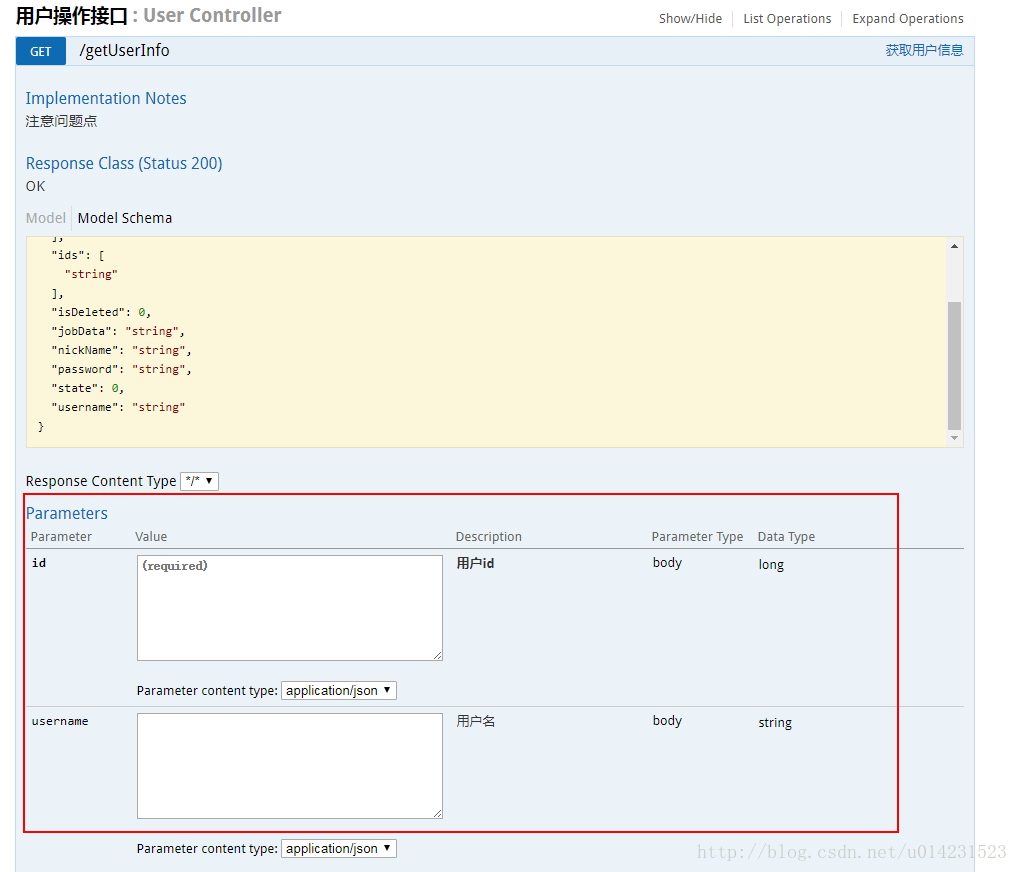
// userService可忽略，是业务逻辑

User user = userService.getUserInfo();

return user;

}

}

效果图：   


**@ApiModel()**用于类 ；表示对类进行说明，用于参数用实体类接收   
value–表示对象名   
description–描述   
都可省略   
**@ApiModelProperty()**用于方法，字段； 表示对model属性的说明或者数据操作更改   
value–字段说明   
name–重写属性名字   
dataType–重写属性类型   
required–是否必填   
example–举例说明   
hidden–隐藏

@ApiModel(value="user对象",description="用户对象user")

public class User implements Serializable{

private static final long serialVersionUID = 1L;

@ApiModelProperty(value="用户名",name="username",example="xingguo")

private String username;

@ApiModelProperty(value="状态",name="state",required=true)

private Integer state;

private String password;

private String nickName;

private Integer isDeleted;

@ApiModelProperty(value="id数组",hidden=true)

private String[] ids;

private List<String> idList;

//省略get/set

}

@ApiOperation("更改用户信息")

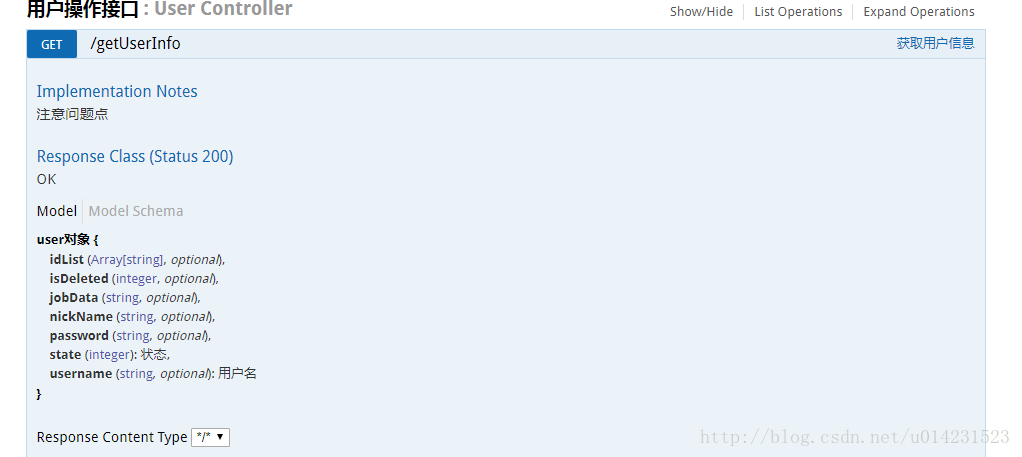
@PostMapping("/updateUserInfo")

public int updateUserInfo(@RequestBody @ApiParam(name="用户对象",value="传入json格式",required=true) User user){

int num = userService.updateUserInfo(user);

return num;

}

效果图：   




**@ApiIgnore()**用于类或者方法上，可以不被swagger显示在页面上   
比较简单, 这里不做举例

**@ApiImplicitParam()** 用于方法   
表示单独的请求参数   
**@ApiImplicitParams()** 用于方法，包含多个 @ApiImplicitParam   
name–参数ming   
value–参数说明   
dataType–数据类型   
paramType–参数类型   
example–举例说明

@ApiOperation("查询测试")

@GetMapping("select")

//@ApiImplicitParam(name="name",value="用户名",dataType="String", paramType = "query")

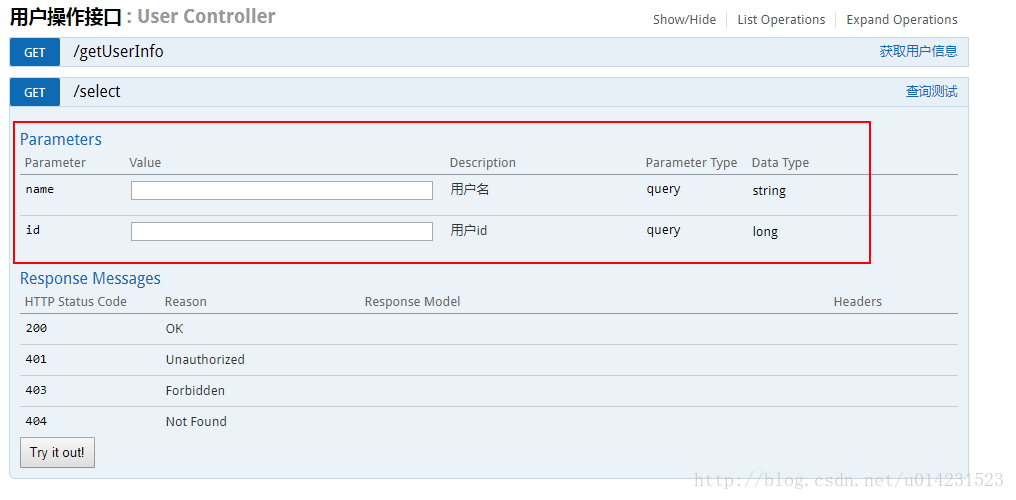
@ApiImplicitParams({

@ApiImplicitParam(name="name",value="用户名",dataType="string", paramType = "query",example="xingguo"),

@ApiImplicitParam(name="id",value="用户id",dataType="long", paramType = "query")})

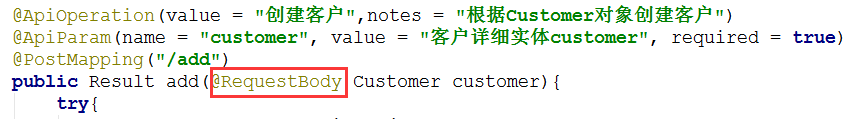
public void select(){

}

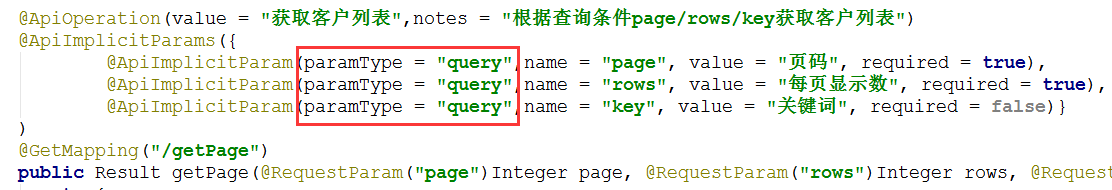
效果图：   


使用过程中对swagger2进行总结

添加客户对象时必须使用@RequestBody，否则swagger2无法获取实例对象



多条件查询时要使用“query”，不然获取不到参数



@ApiImplicitParam与@ApiParam的区别

一个加paramType = **"query"，一个不加**

使用@ApiImplicitParam(paramType = **"query"**,name = **"id"**, value = **"客户编号id"**, required = **true**)

使用@ApiParam(name = **"id"**, value = **"客户编号id"**, required = **true**)

@ApiImplicitParam加类型可能导致类型不符(使用Integer时遇到该问题，去掉类型正常)

