**swagger2**

# 解决的痛点

大家所工作的项目组可能会有前端团队和后端团队，后端团队可能又有不同群组，这些若干个团队日常会各自开发各自的系统，然后对外部提供各类接口。

问题是：

这些接口没有一个统一查看的地方。

这些接口在开发过程中经常被调整，无法实时告知使用者关于接口的变更。

需要做个文档专门管理这些接口。

没有一个统一的调试工具来调试这些接口。

# 什么是swagger2

Swagger 是一款RESTFUL接口的文档在线自动生成+功能测试功能软件。它能帮助我们生成接口文档：实时对外开放我们的api接口。

# 如何使用swagger2

## 整合安装配置

在准备集成swagger2的项目中修改如下几项

### pom文件

将下面的文件复制黏贴到pom文件中。

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.4.0</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.4.0</version>

</dependency>

### 新建初始化类



里边有注释，按照注释修改。

### vo实体类

成员变量上添加注解，例：

@ApiModelProperty(value = "部门列表")

private Map<String, String> deptList;

### controller实体类

类名上追加

@Api("TestController-api")

然后在方法体上追加

@ApiOperation(value = "测试swagger-value", notes = "测试swagger-notes")

## 测试

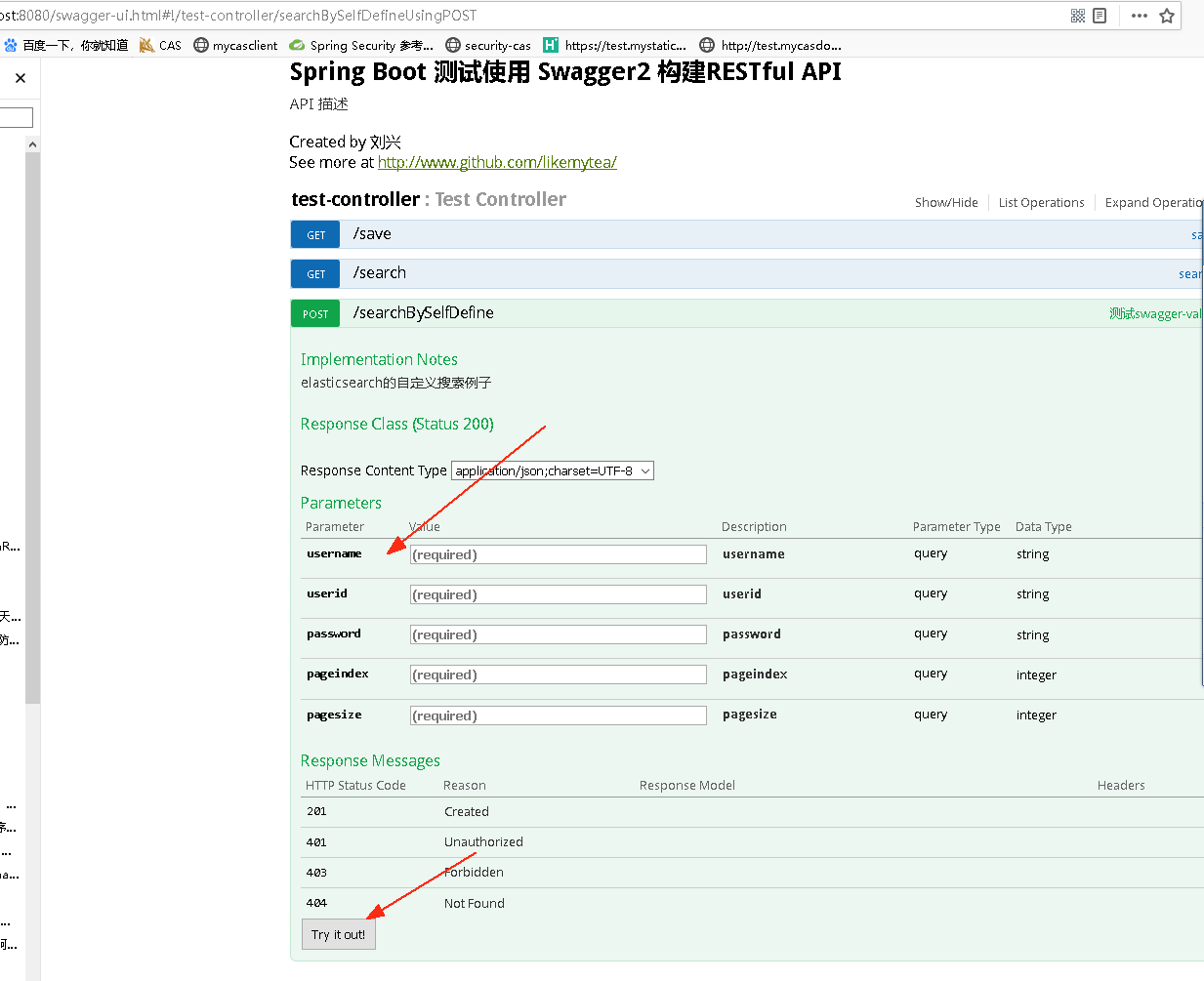
启动项目并访问如下路径即可查看实现效果

<http://localhost:8080/>你的contextpath/swagger-ui.html

打开后的页面是这样的：

第一个箭头：是这个接口需要传入的参数

第二个箭头：点击后可测试此接口



## demo

下载地址：<https://github.com/likemytea/elasticsearch> ，导入到eclipse即可启动测试。

这个demo也是 elasticsearch的客户端调用实例：

http://localhost:8080/searchBySelfDefine?username=李名

&userid=&password=&pageindex=0&pagesize=100

## 注解说明

# Swagger2 注解说明

1. \*\*@Api\*\*：用在请求的类上，主要对类的说明

tags="说明该类的作用，可以在UI界面上看到的注解"

value="该参数没什么意义，在UI界面上也看到，所以不需要配置"

示例：

@Api("用户信息管理")

@RestController

@RequestMapping("/user/\*")

public class UserController {

}

2. \*\*@ApiOperation\*\*：用在请求的方法上，说明方法的用途、作用

value="说明方法的用途、作用"

notes="方法的备注说明"

示例：

@ApiOperation("获取用户列表")

@GetMapping("list")

public List userList() {

return userList;

}

3. \*\*@ApiImplicitParams\*\*：用在请求的方法上，表示一组参数说明

4. \*\*@ApiImplicitParam\*\*：用在@ApiImplicitParams注解中，指定一个请求参数的各个方面

name：参数名

value：参数的汉字说明、解释

required：参数是否必须传

paramType：参数放在哪个地方

· header --> 请求参数的获取：@RequestHeader

· query --> 请求参数的获取：@RequestParam

· path（用于restful接口）--> 请求参数的获取：@PathVariable

· body（不常用）

· form（不常用）

dataType：参数类型，默认String，其它值dataType="Integer"

defaultValue：参数的默认值

示例：

@ApiImplicitParams({

@ApiImplicitParam(name="ajbh",value="案件编号",required=true,paramType="form"),

@ApiImplicitParam(name="name",value="名称",required=true,paramType="form"),

@ApiImplicitParam(name="age",value="年龄",required=true,paramType="form",dataType="Integer")

})

5. \*\*@ApiResponses\*\*：用在请求的方法上，表示一组响应

6. \*\*@ApiResponse\*\*：用在@ApiResponses中，一般用于表达一个错误的响应信息

code：数字，例如400

message：信息，例如"请求参数没填好"

response：抛出异常的类

示例：

@ApiOperation(value = "select1请求",notes = "多个参数，多种的查询参数类型")

@ApiResponses({

@ApiResponse(code=400,message="请求参数没填好"),

@ApiResponse(code=404,message="请求路径没有或页面跳转路径不对")

})

7. \*\*@ApiModel\*\*：用于响应类上，表示一个返回响应数据的信息

这种一般用在post创建的时候，使用@RequestBody这样的场景，

请求参数无法使用@ApiImplicitParam注解进行描述的时

8. \*\*@ApiModelProperty\*\*：用在属性上，描述响应类的属性

示例：

@ApiModel(description= "返回响应数据")

public class RestMessage implements Serializable{

@ApiModelProperty(value = "是否成功")

private boolean success=true;

@ApiModelProperty(value = "返回对象")

private Object data;

@ApiModelProperty(value = "错误编号")

private Integer errCode;

@ApiModelProperty(value = "错误信息")

private String message;

}