

## 尼斯湖水鬼的博客

SpringBoot开发, SpringCloud开发, Java开发



RSS订阅

### SpringBoot开发详解 (八) -- 使用Swagger2构建API文档

2017年05月21日 23:13:41

阅读数: 1059

5

目录

收藏

## API文档

文档在开发中的价值与作用:

评论

作为一个开发人员, 平时看得最多的恐怕就是各式各样的文档了。并且在开发中我们也避免不了的需要自己去书写文档, 比如作为后台开发人员, 我们书写最多的应该就是接口文档了。前端人员会按照我们给出的文档来进行前端开发, 并且按照文档细节来构建不同的传输协议, 对象定义, 字段解析等等。

微信

我曾长时间的使用EXCEL来书写接口文档, 可是逐渐的也暴露出一些问题:

微博

- 接口过多, 每一个接口需要一个入参定义以及出参定义, 不同的请求类型。书写过多, 浪费时间。
- 输出参数类型定义与文档不符, 前端外无法解析, 耗费时间。
- 接口修改需要同步修改文档, 重复工作。
- 多人开发文档常常造成冲突。

为了解决以上所描述的问题, 我们引入了Swagger2, 它可以减少我们书写文档的工作量, 并且可以保持代码与文档的一致性。同时, 可以通过页面测试来直接挑刺接口。说了这么多的好处, 我们来看看我们应该如何在SpringBoot中如何使用Swagger2吧。

## swagger

default (/v2/api-docs)  Explore

### API文档

API使用即参数定义

Created by ZZP

result-controller : Result Controller

Show/Hide | List Operations | Expand Operations

round-2-controller : Round 2 Controller

Show/Hide | List Operations | Expand Operations

user-controller : User Controller

Show/Hide | List Operations | Expand Operations

POST /user/createUser

createUser

POST /user/createUser2

createUser2

POST /user/createUserByMap

createUserByMap

DELETE /user/deleteUserByUserId/{id}

deleteUserById

GET /user/getUser/{id}

getUser

PUT /user/updateUser/{id}

updateUser

xml-controller : XML Controller

Show/Hide | List Operations | Expand Operations

[ BASE URL: /, API VERSION: 0.1 ]

[http://blog.csdn.net/qq\\_31001665](http://blog.csdn.net/qq_31001665)

这样的文档是不是看起来很简单明了呢。

引入Swagger2依赖:

```

1 <!--swagger2依赖, 构建API文档-->
2 <dependency>
3   <groupId>io.springfox</groupId>
4   <artifactId>springfox-swagger2</artifactId>
5   <version>2.2.2</version>
6 </dependency>
7 <dependency>
8   <groupId>io.springfox</groupId>
9   <artifactId>springfox-swagger-ui</artifactId>
10  <version>2.2.2</version>
11 </dependency>
```

### 个人资料



尼斯湖水鬼

[关注](#)

原创  
12

粉丝  
185

喜欢  
23

评论  
43

等级: 博客 4 访问: 13万+

积分: 1090 排名: 4万+



### 呼叫中心系统



### 最新文章

[SpringBoot开发详解 \(十二\) -- SpringBoot中执行定时任务](#)

[SpringBoot开发详解 \(十一\) -- Redis在SpringBoot中的整合](#)

[SpringBoot开发详解 \(十\) -- 使用JPA访问数据库下篇及使用Page进行数据分页](#)

[SpringBoot开发详解 \(九\) -- 使用JPA访问数据库上篇](#)

[SpringBoot开发详解 \(七\) -- Mybatis整合Spring Boot](#)

### 个人分类

spring boot

12篇

### 归档

2017年7月

2篇

2017年6月

1篇

2017年5月

5篇

2017年4月

3篇

2017年2月

1篇

### 热门文章

[SpringBoot开发详解 \(五\) --Controller接收参数以及参数校验](#)

阅读量: 55581

[SpringBoot开发详解 \(三\) --SpringBoot配置文件YML注意事项](#)

阅读量: 17424

[SpringBoot开发详解 \(十二\) -- SpringBoot中执行定时任务](#)

阅读量: 12598

在引入Swagger2的依赖后，我们需要创建一个配置类。这个配置类应该是和你的启动类是同级的，这里，我直接在启动类中进行代码书写了（就是这么懒.....）

```
1  /**
2   * 启动类
3   */
4  @RequestMapping(value = "/")
5  @RestController
6  @SpringBootApplication
7  @MapperScan(basePackages = "com.zzp.dao")
8  @Configuration
9  @EnableSwagger2
10 public class Round1Application {
11
12     @RequestMapping(value = "/",method = RequestMethod.GET)
13     public String helloWorld(){
14         return "Hello World";
15     }
16     public static void main(String[] args) {
17         SpringApplication.run(Round1Application.class, args);
18     }
19
20
21     @Bean
22     public Docket createApi(){
23         return new Docket(DocumentationType.SWAGGER_2).apiInfo(apiInfo()).select()
24             .apis(RequestHandlerSelectors.basePackage("com.zzp.controller"))
25             .paths(PathSelectors.any())
26             .build();
27     }
28
29     private ApiInfo apiInfo(){
30         return new ApiInfoBuilder()
31             .title("API文档")
32             .description("API使用即参数定义")
33             .termsOfServiceUrl("http://blog.csdn.net/qq_31001665")
34             .contact("ZZP")
35             .version("0.1")
36             .build();
37     }
38 }
```

我们通过@Configuration注解，让Spring启动时加载配置，通过@EnableSwagger2开启Swagger。话说一般开庆某个功能都是@EnableXXX的。

apiInfo中的内容是文档的一些基本配置信息，而createApi中则是确定扫面哪些包下的接口形成文档，以及文档展示哪些信息。注意这里是一个Docket的bean。

完成配置类的书写后我们已经可以直接访问我们的文档了，启动项目，访问

<http://localhost:9090/swagger-ui.html>

是不是看见如下内容了呢：

The screenshot shows the Swagger UI homepage for the 'Round1Application'. At the top, there's a navigation bar with links for 'swagger' and 'Explore'. Below the navigation, the title 'API文档' (API Documentation) is displayed, along with a note 'API使用即参数定义' (API usage is parameter definition). A section titled 'Created by ZZP' is present. The main content area lists four controllers: 'result-controller : Result Controller', 'round-2-controller : Round 2 Controller', 'user-controller : User Controller', and 'xml-controller : XML Controller'. Each controller entry includes three buttons: 'Show/Hide', 'List Operations', and 'Expand Operations'. At the bottom of the page, a note indicates '[ BASE URL: /, API VERSION: 0.1 ]'.

SpringBoot开发详解（六）-- 异常统一管理

以及AOP的使用

阅读量：9583

SpringBoot开发详解（十一）-- Redis在SpringBoot中的整合

阅读量：8411

### 最新评论

SpringBoot开发详解（十一）...

xia\_zai\_pin\_dao\_：按照博主的思路写下来后，程序运行是正确的。但是在redis的客户端却是乱码，而且看不到值。但是用代...

SpringBoot开发详解（六）...

qq\_35981283：很好的文章

SpringBoot开发详解（一）...

qq\_31211685：很详细，赞一个

SpringBoot开发详解（六）...

weixin\_42096420：楼主写得很好，感谢，接下来的我都要看看，太棒了

SpringBoot开发详解（五）...

sss1342746626：感谢楼主。。终于发现一个接收数据好用的方法了。。。赞



郑州招聘会

### 联系我们



请扫描二维码联系客服

✉ webmaster@csdn.net

☎ 400-660-0108

✉ QQ客服 ✉ 客服论坛

关闭 报错 广告服务 反馈 反馈

没错这就是我们的四个controller，每个点击开来就可以看到我们的每一个接口了。不过没有中文注释的接口始终对于用户不太友好。我们接下来的工作就是给每一个接口添加注释。这里我们使用@ApiOperation注解来注释我们接口的一些基本

信息，使用@ApiImplicitParams, @ApiImplicitParam注解来注释我们的入参信息。

```
1  @RequestMapping("/user")
2  @RestController
3  public class UserController {
4
5      @Autowired
6      private UserService userService;
7
8      /**
9       * 添加用户
10      * @param tel 注册手机号
11      * @param pwd 设置密码
12      */
13     @ApiOperation(value = "创建用户",notes = "使用手机以及密码初始化用户信息")
14     @ApiImplicitParams({
15         @ApiImplicitParam(name = "tel",value = "用户手机号",required = true,dataType = "String"),
16         @ApiImplicitParam(name = "pwd",value = "用户初始密码",required = true,dataType = "String")
17     })
18     @PostMapping("/createUser")
19     public void createUser(@RequestParam("tel") String tel, @RequestParam("pwd") String pwd){
20         userService.createUser(tel,pwd);
21     }
22
23     /**
24      * 添加用户2
25      * @param userInfo
26      * @Valid添加表单验证, BindingResult获取验证结果
27      */
28     @ApiOperation(value = "创建用户v2版本",notes = "使用UserInfo对象初始化用户信息")
29     @ApiImplicitParam(name = "userInfo",value = "用户对象",required = true,dataType = "UserInfo")
30     @PostMapping("/createUser2")
31     public String createUser2(@Valid UserInfo userInfo, BindingResult bindingResult){
32         if (bindingResult.hasErrors()){
33             return bindingResult.getFieldError().getDefaultValue();
34         }
35         userService.createUser(userInfo.getTel(),userInfo.getPassWord());
36         return "OK";
37     }
38
39     /**
40      * 更新用户信息
41      * @param user_id 用户ID
42      * @param nickName 昵称
43      */
44     @PutMapping("/updateUser/{id}")
45     public void updateUser(@PathVariable("id") String user_id, @RequestParam("nickName") String nickName){
46         userService.updateUser(user_id,nickName);
47     }
48
49     /**
50      * 获取用户信息
51      * @param id 用户Id
52      * @return
53      */
54     @GetMapping("/getUser/{id}")
55     public UserInfo getUser(@PathVariable("id") Integer id){
56         return userService.getUser(id);
57     }
58
59     /**
60      * 删除用户
61      * @param id
62      */
63     @DeleteMapping("/deleteUserById/{id}")
64     public void deleteUserById(@PathVariable("id") Integer id){
65         userService.deleteUserById(id);
66     }
67
68     /**
69      * 使用@RequestBody获取参数, 用map类型接收, 再取出
70      * @param reqMap
71      */
72     @PostMapping("/createUserByMap")
73     public void createUserByMap(@RequestBody Map<String, Object> reqMap){
```

```
74     String tel = reqMap.get("tel").toString();
75     String pwd = reqMap.get("pwd").toString();
76     userService.createUser(tel,pwd);
77 }
78 }

1  @RestController
2  @RequestMapping("/xml")
3  public class XMLController {
4
5      @Autowired
6      private XMLService service;
7
8      @Autowired
9      private ExceptionHandle handle;
10
11     /**
12      * 更新用户信息
13      * @param user_id 用户ID
14      * @param nickName 昵称
15      */
16     @ApiOperation(value = "更新用户信息",notes = "更新用户昵称")
17     @ApiImplicitParams({
18         @ApiImplicitParam(name = "id",value = "用户id",required = true,dataType = "String"
19         @ApiImplicitParam(name = "nickName",value = "用户昵称",required = true,dataType =
20     })
21     @PutMapping("/updateUser/{id}")
22     public Result updateUser(@PathVariable("id") String user_id, @RequestParam("nickName") St
23         Result result = ResultUtil.success();
24         try {
25             service.updateUser(user_id,nickName);
26         }catch (Exception e){
27             result = handle.exceptionGet(e);
28         }
29         return result;
30 //        service.updateUser(user_id,nickName);
31     }
32
33     /**
34      * 获取用户信息
35      * @param id 用户Id
36      * @return
37      */
38     @ApiOperation(value = "获取用户信息",notes = "返回用户信息")
39     @ApiImplicitParam(name = "id",value = "用户id",required = true,dataType = "Integer",paramType =
40     @GetMapping("/getUser/{id}")
41     public Result getUser(@PathVariable("id") Integer id){
42         Result result = ResultUtil.success();
43         try {
44             result.setData(service.getUser(id));
45         }catch (Exception e){
46             result = handle.exceptionGet(e);
47         }
48         return result;
49 //        return service.getUser(id);
50     }
51
52     /**
53      * 删除用户
54      * @param tel
55      */
56     @ApiOperation(value = "删除用户",notes = "根据用户id删除用户")
57     @ApiImplicitParam(name = "id",value = "用户id",required = true,dataType = "Integer")
58     @DeleteMapping("/deleteUserByUserId/{tel}")
59     public Result deleteUserByUserId(@PathVariable("tel") String tel{
60         Result result = ResultUtil.success();
61         try {
62             UserInfo user = new UserInfo();
63             user.setTel(tel);
64             service.deleteUserByUserId(user);
65         }catch (Exception e){
66             result = handle.exceptionGet(e);
67         }
68         return result;
69 //        UserInfo user = new UserInfo();
```

```

70 //         user.setTel(tel);
71 //         service.deleteUserByUserId(user);
72 }
73
74 /**
75 * 使用@RequestBody获取参数，用map类型接收，再取出
76 * @param reqMap
77 */
78 @ApiOperation(value = "创建用户v3版本",notes = "返回用户信息")
79 @ApiImplicitParam(name = "Map",value = "map集合",required = true,dataType = "Map")
80 @PostMapping("/createUserByMap")
81 public Result createUserByMap(@RequestBody Map<String, Object> reqMap){
82     Result result = ResultUtil.success();
83     try {
84         service.createUser(reqMap);
85     }catch (Exception e){
86         result = handle.exceptionGet(e);
87     }
88     return result;
89 //     service.createUser(reqMap);
90 }
91
92
93 }

```

这里我给两个controller都加入了直接配置，因为在使用mybatis后，spring会默认使用mybatis配置，jdbc链接数据库会报错：Invalid bound statement (not found): com.zzp.dao.UserInfoMapper.getUser。大家在做项目时，一定要使用一种方法来链接数据库，不然各种错误会一直困扰着你。

我们完成上述代码的添加后就可以在刚才的地址中看到我们配置的接口信息了

**PUT /xml/updateUser/{id}** 更新用户信息

**Implementation Notes**  
更新用户名称

**Response Class (Status 200)**  
Model Model Schema

```
{
  "data": {},
  "msg": "string",
  "status": 0
}
```

**Response Content Type** \*/\*

**Parameters**

Parameter	Value	Description	Parameter Type	Data Type
<b>id</b>	(required)	用户id	body	String
Parameter content type: application/json				
<b>nickName</b>	(required)	用户昵称	body	String
Parameter content type: application/json				

**Response Messages**

HTTP Status Code	Reason	Response Model	Headers
201	Created		<a href="http://blog.csdn.net/qq_31001665">http://blog.csdn.net/qq_31001665</a>

并且我们可以点击 Try it out! 来测试我们的接口了。我们这里使用Get方法来获取用户信息的接口测试一下，发现我们已经获取了用户信息，要注意的是如果你也是直接通过URL路径中获取参数，那需要添加paramType = "path"这个参数，不然无法获取到id值的。

HTTP Status Code	Reason	Response Model	Headers
401	Unauthorized		
403	Forbidden		
404	Not Found		

[Try it out!](#) [Hide Response](#)

**Curl**

```
curl -X GET --header "Accept: */*" "http://localhost:9090/xml/getUser/13"
```

**Request URL**

```
http://localhost:9090/xml/getUser/13
```

**Response Body**

```
{
  "status": 0,
  "msg": "success",
  "data": {
    "tel": "13800000000",
    "nickName": "pppzzz",
    "passWord": "827ccb0eea8a706c4c34a16891f84e7b"
  }
}
```

**Response Code**

```
200
```

**Response Headers**

```
{
  "date": "Sun, 21 May 2017 14:17:17 GMT",
  "transfer-encoding": "chunked",
  "content-type": "application/json;charset=UTF-8"
}
```

## Swagger2写在最后的话：

相比较之前我们使用EXCEL来书写接口文档，Swagger2的确方便了许多，并且书写量也大大减少，看似使用Swagger2来对API文档进行管理是一个不错的选择。其实不然（逃），因为Swagger2对于代码的污染和侵入性我认为太大了。并且你应该也发现了，当你使用Map作为参数时，Swagger2很难对Map内的每一个参数进行说明（你可以写超多文字描述）。所以在实际开发中，Swagger2的使用并不是很多，如果你是个人开发者，那使用简单的Swagger2的确可以满足需求。如果你构建的是庞大的接口系统时。我的建议是使用Swagger2和MarkDown配合完成文档的书写。MarkDown书写方便高效，语法简单，是我们值得学习的。而Swagger2提供接口调试以及字段名匹配来保证入参与出参的一致性。

以上所有的代码我已经上传到GitHub

round1-springboot

如果心急的小伙伴也可以去clone我已经完成的项目，这个项目中把一些常用功能都写了，并且都写注释啦！！！

# MySpringBoot

版权声明：本文为博主原创文章，未经博主允许不得转载。[https://blog.csdn.net/qq\\_31001665/article/details/72616703](https://blog.csdn.net/qq_31001665/article/details/72616703)

文章标签: [api](#) [spring-boot](#) [swagger-json](#) [Api文档](#)

个人分类: spring boot



那都不是事！学会AI这些技能，让你叱咤人工智能领域！

时代趋势，顺流而上！掌握人工智能，避免就业危机！成为稀缺人才我们有方法...

12523

查看更多

想对作者说点什么？  
我来说一句

菜鸟上路 Spring Boot+Spring data jpa+swagger 做的增 删 改 查

1.数据库用的是mysql数据库，项目整体结构如下，项目中Hystrix和Eureka可以看看，新手不晓得怎么上传项目文件 研究下后续给补上 2.用的maven 这里贴出maven 的pom配置...

 qq\_24709699 2017-07-05 13:34:49 阅读数: 472

springboot+mybatis、JPA+swagger2完美集成

Java上常用的几个框架 **SpringBoot**、**Mybatis**、**JPA**、**SpringData**、**Swagger**，它们各自的功能，以及它们之间的关系，我依着我对这些框架的理解，以最简单精炼...

qq\_28125445 2017-02-17 13:48:47 阅读数: 2676

## Apache shiro+springmvc+springdata+jpa+swagger(零配置文件使用)

apache shiro 简介 shiro是Apache退出的认证授权框架,帮我们封装了认证授权接口,我们只需要实现这些接口即可,不需要关心回话管理授权及拦截等功能. springdata 简介 ...

huabenye 2016-07-29 11:00:14 阅读数: 3419

## 使用springfox+swagger2书写API文档

主要介绍springfox自动生成API文档的配置与使用并结合swagger2展示书写的API, 着重介绍springMVC中如何使用配置、使用springfox ...

u012476983 2017-01-05 10:37:31 阅读数: 32002

## spring-boot+mybatis+pagehelper+Swagger2构建RESTful API

spring-boot+jpa 确实十分强大 个人也很喜欢jpa中的方法名称推断因为项目要用到spring-boot+mybatis 所以就有了这篇文章 简单的把例子在这里贴出来 方便以后查看代码已经...

zhugeyangyang1994 2016-07-26 18:25:41 阅读数: 5278

## springboot整合swagger UI 、spring-data-JPA

本博客为个人所写, 有不对地方请大家指正! 1.首先目录结构 2.pom.xml

lijunfan\_rh 2017-01-17 10:41:14 阅读数: 1318

## Springboot+Swagger2

Swagger为大家节省了很多时间,写完了接口,相信大家已经身心疲惫了,可是又要面对繁琐无味的接口文档,这个时候Swagger出现了。 Swagger能成为最受欢迎的REST APIs文档生成工具...

u013128651 2018-01-25 20:17:08 阅读数: 183

## Idea+SpringBoot+Mybatis+Mysql+Gradle+Swagger2 - --动态查询

开发工具: Idea (2017.1.5) jdk1.8 使用框架, 数据库, jar库依赖: SpringBoot+Mybatis, Mysql, Gradle4.0 模仿需求: 图片的上传, 下载, 商品类型的...

Mynah886 2017-07-20 15:35:42 阅读数: 473

## SpringBoot+SpringData实现SpringMVC简单项目

最近一段时间在学习《Spring in action》这本书。为什么要使用Spring? 因为可以减少Java开发的复杂性。为什么要使用Spring Boot? 为了简化Spring应用的搭建和开发过...

mrtif 2016-08-25 16:25:01 阅读数: 1027

## Spring Boot中使用Swagger2构建RESTful API文档

转载声明: 商业转载请联系作者获得授权,非商业转载请注明出处 © wekri 随着时间推移, 不断修改接口实现的时候都必须同步修改接口文档, 而文档与代码又处于两个不同的媒介, 除非有严格的管理机制...

Eacter 2017-02-23 15:01:25 阅读数: 4081

## Spring Boot 使用Swagger2自动生成RESTful API文档

几个简单的步骤, 就可以在Spring Boot中配置Swagger2来实现API文档自动生成。

ClementAD 2017-02-15 17:55:03 阅读数: 2864

## Spring Boot学习笔记 - 整合Swagger2自动生成RESTful API文档

在App后端开发中经常需要对移动客户端 (Android、iOS) 提供RESTful API接口, 在后期版本快速迭代的过程中, 修改接口实现的时候都必须同步修改接口文档, 而文档与代码又处于两个不同的媒介, ...

FX\_SKY 2017-01-04 19:28:17 阅读数: 8291

## Spring Boot中使用Swagger2构建RESTful APIs

构建RESTful API适配多终端

zhouseawater 2016-12-30 13:36:03 阅读数: 3437

### 高效、准确、低成本的文字识别api

证件、文件一键扫描，准确率高达99%。经过海量用户和多种复杂场景考验



### Spring-Boot + Swagger2 自动生成API接口文档

spring-boot作为当前最为流行的Java web开发脚手架，相信越来越多的开发者会使用其来构建企业级的RESTful API接口。这些接口不但会服务于传统的web端（b/s），也会服务于移动端...

amon1991 2017-08-06 15:51:48 阅读数: 4617

## Spring Boot中使用Swagger2构建RESTful APIs (含源码)

Swagger2简介 本次教程是Spring Boot中使用Swagger2构建RESTful APIs Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的...

wenteryan 2017-11-13 14:18:52 阅读数: 366

### Swagger2 生成 Spring Boot API 文档

Swagger2 生成 Spring Boot API 文档 Swagger2 生成 Spring Boot API 文档 POM 文件 代码支持 访问地址 Swagger 是...

programmer\_sean 2017-05-16 11:17:24 阅读数: 1437

## Spring Boot学习(五)之使用Swagger2构建强大的RESTful API文档

随着前后端的分离，接口文档变的尤其重要，今天我们来说一说用SWAGGER2，来风骚的生成api文档。配置很简单，废话少说，直接上代码：pom.xml文件 01 parent> ...

zhaokejin521 2017-11-10 10:08:56 阅读数: 180

### 使用Swagger2构建强大的RESTful API文档

本文将介绍RESTful API的重磅好伙伴Swagger2，它可以轻松组织出强大RESTful API文档。它既可以减少我们创建文档的工作量，同时说明内容又整合入实现代码中，让维护文档和修改代码整合...

cxhtgm 2017-11-27 13:08:37 阅读数: 461

### 高端网站设计

品牌创意 万户网络专注高端网站建设



## SpringBoot整合Swagger自动生成API文档

swagger用于定义API文档。好处：前后端分离开发 API文档非常明确 测试的时候不需要再使用URL输入浏览器的方式来访问Controller 传统的输入URL的测试方式对于post请求的传参比...

minicto 2017-10-11 15:55:22 阅读数: 824

### 打造完美接口文档 - 应用springboot+swagger2编写restFull接口文档

占位

ruglcc 2017-07-26 11:44:00 阅读数: 1297

### Spring Boot Swagger2 构建RESTful API