(19)中华人民共和国国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 109683886 A
(43)申请公布日 2019.04.26

(21)申请号 201811433944.2

(22)申请日 2018.11.28

(71)申请人 国云科技股份有限公司
地址 523808 广东省东莞市松山湖高新技术产业开发区科汇路1号中科院云计算中心19楼

(72)发明人 张志江 季统凯

(74)专利代理机构 广东莞信律师事务所 44332
代理人 陈熙

(51)Int.Cl.
G06F 8/35(2018.01)
G06F 8/41(2018.01)

权利要求书1页 说明书5页 附图1页

(54)发明名称

一种多元化接口文档生成方法

(57)摘要

本发明涉及Java服务端开发技术领域，特别是一种多元化接口文档生成方法。本发明首先要在模块Controller类头部加入@ApiDoc注释，用来将接口分类，然后在接口入口方法那里加入@ApiOperation，接着设置接口地址、名字、描述、权限、入参类、出参类以及header头部类，最后项目启动的时候，会自动扫描所有需要生成接口文档的方法，并且生成定制化的接口文档。本发明解决了接口文档跟随代码实施同步问题；让开发人员、测试人员以及运维人员可以更清晰、更准确地看懂系统自动生成出来的接口文档。

CN 109683886 A

1.一种多元化接口文档生成方法,其特征在于:所述的方法包括如下步骤:

步骤1:基于SpringMvc在接口类头部加入@ApiDoc注释,标记该类需要被扫描器发现;

步骤2:在需要添加接口文档的Api方法上面加入@ApiOperation注释,用来标记该方法需要添加到接口文档上;

步骤3:在@ApiOperation注释上编写接口URL地址、接口文档的标题、接口对应的权限标记、描述,指定入参类、出参类以及header头部类,这些属性都让扫描器录入并且展示到接口文档上;

步骤4:在项目启动方法中,加入ApiLogUtil调用方法,开始扫描项目中带有注释标记的类和方法;

步骤5:基于JavaPOI来编辑word文档生成出来的格式,定制接口文档的样式;

步骤6:项目启动,生成接口文档。

2.根据权利要求1所述的方法,其特征在于:

所述@ApiDoc是用来区分接口类别的注释,每一个@ApiDoc可以设置对应的Key和名字,相同的Key对应同一个类别,从而区分开不同类别的接口方法。

3.根据权利要求1所述的方法,其特征在于:

所述扫描器的启动类用来项目启动时,开始扫描接口注释。

4.根据权利要求1所述的方法,其特征在于:所述的方法支持多层定制参数的特征,基于递归遍历方式,先遍历参数类所有属性,如果属性值不是基础类型,则将该对象中的属性继续遍历,直到遍历到基础类型为止;并且将遍历出来的属性都放进参数队列。

5.根据权利要求1所述的方法,其特征在于:

所述的入参类、出参类和header类,自定义参数类上的属性名字、类型、是否可为空和请求例子。

6.根据权利要求1所述的方法,其特征在于:所述的方法中,编辑生成的word模版格式,可以根据需求定制不同样式的接口文档。

一种多元化接口文档生成方法

技术领域

[0001] 本发明涉及Java服务端开发技术领域,特别是一种多元化接口文档生成方法。

背景技术

[0002] 现在的应用系统越来越庞大,系统修复和更新频率越来越多;接口文档的编辑频率也越来越高;如此会带来以下问题:

[0003] 一是出现接口文档和后台代码不同步问题。

[0004] 二是接口文档无分类,查阅困难。

[0005] 三是多层参数无法展示。

[0006] 四是接口文档格式单一。

[0007] 为了节省开发、测试时间,以及运维人员管理接口更加简便;需要一种多元化接口文档生成方法,从而可以高效、便捷地开发,应对更加庞大的业务系统。

发明内容

[0008] 本发明解决的技术问题在于提供一种多元化接口文档生成方法;解决后台接口文档管理问题。

[0009] 本发明解决上述技术问题的技术方案是:

[0010] 所述的方法包括如下步骤:

[0011] 步骤1:基于SpringMvc在接口类头部加入@ApiDoc注释,标记该类需要被扫描器发现;

[0012] 步骤2:在需要添加接口文档的Api方法上面加入@ApiOperation注释,用来标记该方法需要添加到接口文档上;

[0013] 步骤3:在@ApiOperation注释上编写接口URL地址、接口文档的标题、接口对应的权限标记、描述,指定入参类、出参类以及header头部类,这些属性都让扫描器录入并且展示到接口文档上;

[0014] 步骤4:在项目启动方法中,加入ApiLogUtil调用方法,开始扫描项目中带有注释标记的类和方法;

[0015] 步骤5:基于JavaPOI来编辑word文档生成出来的格式,定制接口文档的样式;

[0016] 步骤6:项目启动,生成接口文档。

[0017] 所述@ApiDoc是用来区分接口类别的注释,每一个@ApiDoc可以设置对应的Key和名字,相同的Key对应同一个类别,从而区分开不同类别的接口方法。

[0018] 所述扫描器的启动类用来项目启动时,开始扫描接口注释。

[0019] 所述的方法支持多层定制参数的特征,基于递归遍历方式,先遍历参数类所有属性,如果属性值不是基础类型,则将该对象中的属性继续遍历,直到遍历到基础类型为止;并且将遍历出来的属性都放进参数队列。

[0020] 所述的入参类、出参类和header类,自定义参数类上的属性名字、类型、是否可为

空和请求例子。

[0021] 所述的方法中，编辑生成的word模版格式，可以根据需求定制不同样式的接口文档。

[0022] 本发明通过Java注释和递归遍历方式，实现接口文档多元化生成，可以支持更多不同场景的接口文档输出，从而也实现接口权限控制。本发明不仅解决了接口文档跟随代码实施同步问题，还打破了业界单一生成接口文档的方法，具有以下效果：

[0023] 1.可以根据不同功能来对接口进行分类展示。

[0024] 2.支持根据客户需求定制不同格式的接口文档模版。

[0025] 3.支持多层定制参数类，可以更加详细地显示嵌套参数类的属性。

[0026] 4.支持接口header参数类生成，目前业界暂无该方案。

[0027] 5.支持权限控制，让开发人员对接口权限一目了然。

[0028] 6.支持restful接口文档生成。

[0029] 让开发人员，测试人员以及运维人员可以更清晰，更准确地看懂系统自动生成出来的接口文档。

附图说明

[0030] 下面结合附图对本发明进一步说明：

[0031] 图1为本发明方法流程图。

具体实施方式

[0032] 见图1所示，本发明的流程如下：

[0033] 步骤1:基于SpringMvc在接口类头部加入@ApiDoc注释，标记该类需要被扫描器发现。

[0034] 步骤2:在需要添加接口文档的Api方法上面加入@ApiOperation注释，用来标记该方法需要添加到接口文档上。

[0035] 步骤3:在@ApiOperation注释上编写接口URL地址、接口文档的标题、接口对应的权限标记、描述、指定入参类、出参类以及header头部类，这些属性都可以让扫描器录入并且展示到接口文档上。

[0036] 步骤4:在项目启动方法中，加入ApiLogUtil调用方法，开始扫描项目中带有注释标记的类和方法。

[0037] 步骤5:基于JavaPOI来编辑word文档生成出来的格式，定制接口文档的样式。

[0038] 步骤6:项目启动，生成接口文档。

[0039] 本发明可以将所有接口按照接口类来分类，支持定制格式生成接口文档，支持权限控制，支持多层定制参数类，支持restful接口文档生成。

[0040] @ApiDoc是用来区分接口类别的注释，每一个@ApiDoc可以设置对应的Key和名字，相同的Key对应同一个类别，从而区分开不同类别的接口方法；

[0041] @ApiOperation是用来配置接口相关属性，为了让接口文档清晰可见，但是又不想改动原有业务代码，本发明可以在@ApiOperation注释中指定入参类、出参类以及header头部类，从而达到接口文档代码不会影响到后台功能；

[0042] restful是一种Api接口协议，本发明支持指定body参数类，URL参数类以及请求方法等，可以同时支持多种数据请求协议；

[0043] 扫描器启动类是用来项目启动的时候，开始扫描接口注释的方法；

[0044] 本发明支持多层定制参数的特征，基于递归遍历方式，先遍历参数类所有属性，如果属性值不是基础类型的话，会继续将该对象中的属性继续遍历，直到遍历到基础类型为止，并且将遍历出来的属性都放进参数队列，这个是目前业界上没有的。

[0045] 编写的出入参数类和header类，是可以自定义参数类上的属性名字、类型、是否可为空和请求例子，可以让文档更加清晰，也不会侵入业务逻辑代码到项目里面，不影响原有功能。所述的编辑word文档生成类，是编辑生成出来的word模版格式，可以根据客户需求定制不同样式的接口文档。

[0046] 此外，本发明扫描器代码如下：

[0047]

```
Map<String, List<ApiBody>> apisMap = new HashMap<String, List<ApiBody>>();
       Map<String, Object> apiDocs =
SpringUtil.getApplicationContext().getBeansWithAnnotation(ApiDoc.class);
       if (apiDocs != null) {
           for (Object apiDoc : apiDocs.values()) {
               List<ApiBody> apis = new ArrayList<ApiBody>();
               String className = apiDoc.getClass().getName();
               Class<?> apiClass = Class.forName(className);

               ApiDoc doc = apiClass.getAnnotation(ApiDoc.class);
               String docName = doc.name();

               RequestMapping requestClassMap =
apiClass.getAnnotation(RequestMapping.class);
               String prefixUrl = "";
               if (requestClassMap != null && requestClassMap.value() != null &&
requestClassMap.value().length > 0) {
                   prefixUrl =
ApiDocUtil.formatUrl(requestClassMap.value()[0]);
               }

               if (apiClass.getMethods() != null) {
                   for (Method apiMethod : apiClass.getMethods()) {
                       ApiOperation ao =
apiMethod.getAnnotation(ApiOperation.class);
                       if (ao != null) {
                           ApiBody api = new ApiBody();
                           api.setDescription(ao.description());
                           api.setName(ao.name());

                           RequestMapping requestMethodMap =
apiMethod.getAnnotation(RequestMapping.class);
                           if (requestMethodMap != null &&
requestMethodMap.value() != null && requestMethodMap.value().length > 0) {
```

[0048]

```
                        api.setUrl(prefixUrl +
ApiDocUtil.formatUrl(requestMethodMap.value()[0]));
                    }

                    api.setRequestMethod(ao.requestMethod().name());

                    Class<?> requestClass = ao.request();
                    Field[] requestFields =
ReflectionUtils.getDeclaredField(requestClass);
                    if (requestFields != null) {

   List<com.gcloud.multicloud.framework.core.model.ApiModel> amList =
addModel(requestFields, false);

   List<com.gcloud.multicloud.framework.core.model.ApiModel> abList =
addModel(requestFields, true);
                        api.setRequestParam(amList);

                    }

                    Class<?> responseClass = ao.response();
                    Field[] responseFields =
ReflectionUtils.getDeclaredField(responseClass);
                    if (responseFields != null) {

   List<com.gcloud.multicloud.framework.core.model.ApiModel> amList =
addModel(responseFields, false);
                        api.setResponse(amList);
                    }

                    apis.add(api);
                }
            }

        }
        apisMap.put(docName, apis);

    }。
```

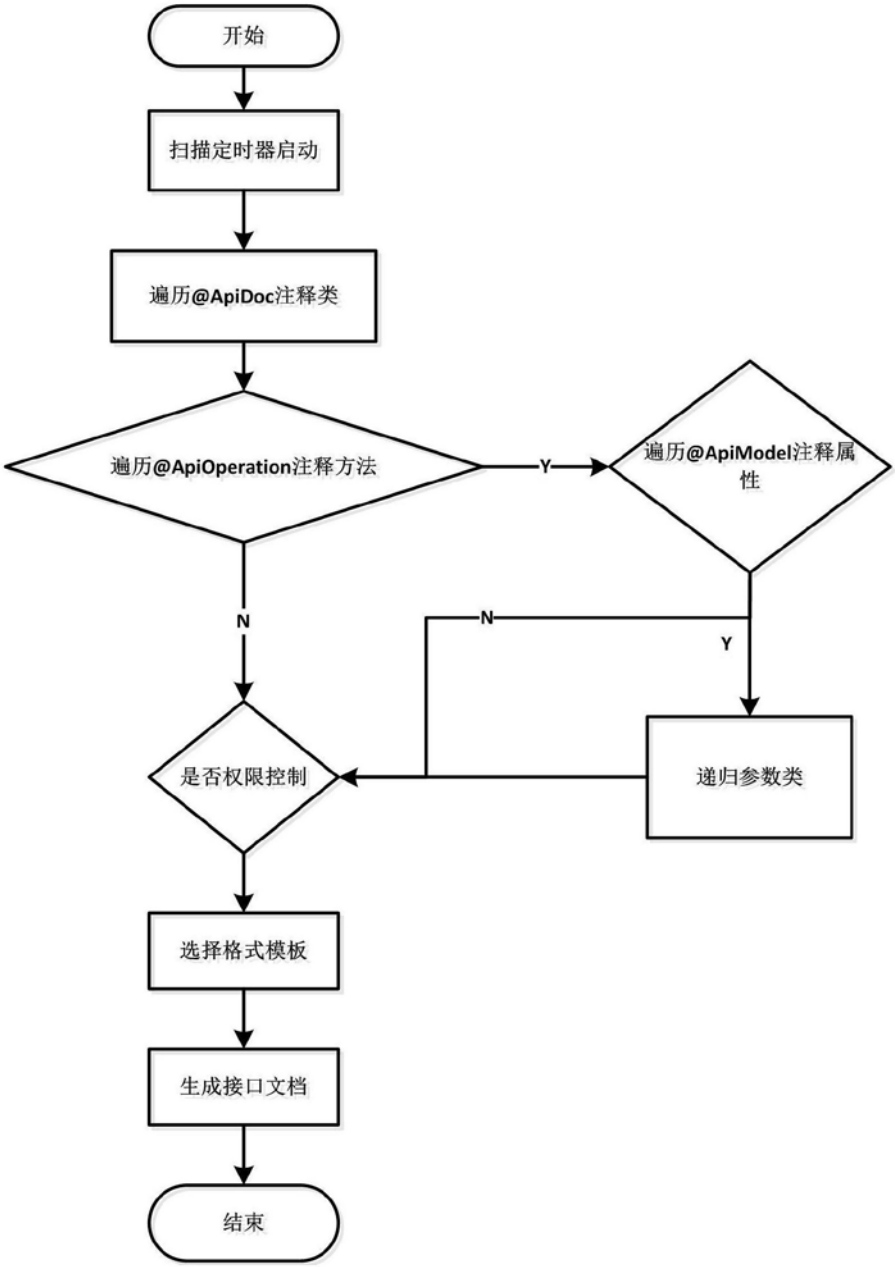
开始
扫描定时器启动
遍历@ApiDoc注释类
遍历@ApiOperation注释方法
Y
遍历@ApiModel注释属性
N
N
Y
是否权限控制
递归参数类
选择格式模板
生成接口文档
结束

图1