Thinkphp集成Swagger-PHP

末日黄昏

目　录

前言

前言

本小册针对swagger-php的常用特性以及实际应用。做了一些参数说明和使用的示例。

环境要求：

- php: >=5.6
- doctrine/annotations: *
- symfony/finder: >=2.2

OpenAP2.0地址：https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#parameterObject

Swagger-PHP地址：http://zircote.com/swagger-php/

安装Swagger

一、Composer安装

本小册针对swagger-php2.0

```
composer require zircote/swagger-php:2.0.*
```

安装完成后，我们就可以编写注释了生成json文件了。

在控制器index中填入以下内容生成swagger.json

```
use Swagger\Annotations as SWG;
class Index
{
     public function index()
     {
        $swagger=\Swagger\scan(__DIR__);
        $res=$swagger->saveAs('./swagger.json');
     }
 }
```

注释部分：

```
/**
* @SWG\Swagger(
* schemes={"http"},
* host="www.tp.com.cn/",
* basePath="/",
* @SWG\Info(
* title="API文档",
* version="1.0.0",
* )
*/)
```

在浏览器中输入地址：

www.tp.com.cn/

swagger-ui配置

Swagger-ui配置

> 去swagger-ui官网下载静态页面，把静态页面放到thinkphp框架目录里，例如public目录。

官网地址：https://swagger.io/tools/swagger-ui/

> 找到swagger-ui下的dist/index.html

```
window.onload = function() {
  // Begin Swagger UI call region
  const ui = SwaggerUIBundle({
    //url:"http://www.tp.com.cn/swagger.json", //单文档地址
      urls:[{url:"http://www.tp.com.cn/swagger.json",name:"前端文档"},{url:"http
://www.tp.com.cn/swagger-1.json",name:"后端文档"}], //开启Topbar插件支持多个文档
    dom_id: '#swagger-ui',
    deepLinking: false,
     validatorUrl: null, //关闭验证规则

    presets: [
      SwaggerUIBundle.presets.apis,
      SwaggerUIStandalonePreset
    ],
    plugins: [
      SwaggerUIBundle.plugins.DownloadUrl
    ],
    layout: "StandaloneLayout"
  })
```

> 打开浏览器访问
>
> http://www.tp.com.cn/swagger-ui/dist/index.html
>
> 更多配置参数访问：
>
> https://github.com/swagger-api/swagger-ui/blob/v3.18.3/docs/usage/configuration.md

配置

配置信息

配置信息

Swagger里使用 `@SWG\Info` 来配置文档的相关信息，例如：

```
/**
 * @SWG\Swagger(
 *     schemes={"http"},                访问协议
 *     host="www.tp.com.cn/",           api链接地址
 *     basePath="/",                    访问路径
 * @SWG\Info(
 *     title="API文档",                  名称
 *     version="1.0.0",                  版本号
 *     description="本文档仅限于测试",   描述
 * )
```

安全配置

全局安全配置

OpenAPI支持的方案包括基本身份验证、API密钥（作为头或作为查询参数）和OAuth2的公共流（隐式、密码、应用程序和访问代码）。在Swagger中使用 `@SWG\SecurityScheme` 定义全局安全配置。

用法如下：

```
/**
* @SWG\SecurityScheme(securityDefinition="apikey",type="apiKey",name="apikey",in="header"),
* /
```

参数	含义
securityDefinition	swager->securityDefinitions数组中的键
type	类型 值为 basic、apiKey、oauth2
name	名称
in	需要API密钥的位置 值为 query、header
flow	OAuth2安全方案使用的流 值为：“implicit”、“password”、“application”或“accesscode”
authorizationUrl	OAuth2授权URL
scopes	OAuth2安全方案的可用范围

apikey示例：

```
/**
*@SWG\SecurityScheme(securityDefinition="apikey",type="apiKey",name="apikey",in="header"),
*/
```

OAuth2示例：

```
/**
*@SWG\SecurityScheme(securityDefinition="auth",type="oauth2",name="auth",scopes={"write:pets"="modify pets in your account","read:pets"="read your pets"},flow="implicit",authorizationUrl="http://swagger.io/api/oauth/dialog")
* /
```

请求

POST请求

POST请求类型

可以通过 `@SWG\Post()` 设置post请求类型的注释

```
/* @SWG\Post(
*     path="/index/register/index",
*     tags={"用户相关"},
*     summary="用户注册",
*     security={{"apikey"={}}},
*     consumes={"application/json"},
*     produces={"application/json"},
*     description="用户请求逻辑，记住用户请求一定要带token",
* ),
```

参数说明：

参数	含义
path	请求路径
tags	用于API文档控制的标记列表。标记可用于按资源或任何其他限定符对操作进行逻辑分组
summary	接口名称
name	名称
consumes	设置parameter请求类型
produces	设置 response 返回类型
description	对接口的详细解释
security	接口的安全设置

apiKey示例：

```
/* *
*@SWG\SecurityScheme(securityDefinition="apikey",type="apiKey",name="apikey",in
="query"),
* @SWG\Post(
*     path="/index/register/index",
*     tags={"用户相关"},
*     summary="用户注册",
*     security={{"apikey"={}}},
* )
 */
```

OAuth2示例：

```
/* *
*@SWG\SecurityScheme(securityDefinition="auth",type="oauth2",name="auth",scopes
={"write:pets"="modify pets in your account","read:pets"="read your pets"},flow
="implicit",authorizationUrl="http://swagger.io/api/oauth/dialog")
* @SWG\Post(
*     path="/index/register/index",
*     tags={"用户相关"},
*     summary="用户注册",
*     security={{auth"={"write:pets","read:pets"}}},
* )
*/
```

> *注意：参数 `security` 定义将覆盖任何已声明的顶级安全性

> 设置多个相同的 `tags` 值，swagger会自动进行逻辑分组

```
/* @SWG\Post(
*     path="/index/index/regiser",
*     tags={"用户相关"},
*     summary="用户注册",
*     consumes={"application/json"},
*     produces={"application/json"},
* @SWG\Response(response="200",description="ok")
* )
*
* @SWG\Post(
*     path="/index/index/user",
*     tags={"用户相关"},
*     summary="用户登录",
*     consumes={"application/json"},
*     produces={"application/json"},
* @SWG\Response(response="200",description="ok")
* )
* /
```

> `@SWG\Get()`，`@SWG\Post()`，`@SWG\Delete()`，`@SWG\Put()` ，4种请求方式里面的参数都相同所以只写了POST的请求示例

Parameter参数设置

Parameter请求类型

参数 `@SWG\Parameter()` 是配合 `@SWG\Post()` , `@SWG\Get()` , `@SWG\Delete()` , `@SWG\Put()` 使用设置请求参数的。

```
/* @SWG\Post(
*     path="/index/register/index",
*     tags={"用户相关"},
*     summary="用户注册",
*     @SWG\Parameter(
*     in="formData",
*     required=false,
*     name="simg",
*     type="file",
*     format="string",
*     description="description",
* ),
* /
```

参数	含义
ref	允许此路径项的外部定义
in	参数的位置，可设置的值为“query”、“header”、“path”、“formData”或“body”
description	描述
required	确定此参数是否是必需的值可设置的值为 true 或 false
type	参数的类型,可设置值是“string”、“number”、“integer”、“boolean”、“array”或“file”
format	类型的扩展格式
items	描述Array上的项目类型
collectionFormat	数组的格式，
default	设置参数的默认值。值的类型取决于定义的类型
maximum	最大长度
maximum	最小长度

下面我们来说下该怎么使用的 `@SWG\Parameter()` 里面的参数：

> 当我们把 `in` 设置为“query”、“header”、“path”、“formData”

示例：

```
/* @SWG\Post(
*     path="/index/register/index",
*     tags={"用户相关"},
*     summary="用户注册",
*     @SWG\Parameter(
*     in="formData",
*     required=false,
*     name="simg",
*     type="file",
*     format="string",
*     description="description",
* ),
* @SWG\Parameter(
*     in="formData",
*     name="num",
*     type="integer",
*     maximum="10",
* ),
*  @SWG\Response(response="200",description="ok")
*
* ),
* /
```

> 需要注意的是参数 `items` 和 `collectionFormat` 只有参数type=“array”才使用：

```
/**  @SWG\Parameter(
*     in="formData",
*     name="ids",
*     type="array",
*     items={"type","string"},
*     collectionFormat="multi"
* ),
* */
```

> 参数 `collectionFormat` 是数组传递到API接口值得格式：csv-逗号分隔值、ssv-空格分隔值、tsv以“%09”分隔值、pipes以“|”分隔值、multi如果 `in` 选择”query“正常接收就好，如果选择“formdata”逗号分隔值

> 当参数 `in` 选择“body”时必须定义 `@SWG\Schema()` 或 `schema` ，通过 `@SWG\Schema()` 自定义一个JSON

```
/*
* @SWG\Post(
*     path="/index/index/regiser",
*     tags={"用户相关"},
*     summary="用户注册",
*     consumes={"application/json"},
*     produces={"application/json"},
*     @SWG\Parameter(
*     in="body",
*     required=true,
*     name="body",
*     description="description",
*     @SWG\Schema(
*              @SWG\Property(
*                  property="firstName",
*                  type="string",
*              ),
*              @SWG\Property(
*                   property="lastName",
*                   type="string"
*              ),
*              @SWG\Property(
*                   property="username",
*                   type="string"
*              )
*          ),
* ),
*   @SWG\Response(response="200",description="ok",)
* )
*/
```

> 关于 `@SWG\Schema()` 的相关介绍可以查看Schema参数说明

Response参数设置

Response请求类型

使用 `@SWG\Response()` 定义response

```
/**  @SWG\Response(response="200",description="ok",
*     @SWG\Schema(
*               @SWG\Property(
*                   property="code",
*                   type="number",
*               ),
*               @SWG\Property(
*                    property="data",
*                    type="object",
*               ),
*               @SWG\Property(
*                    property="msg",
*                    type="string"
*               )
*     ))
*     )
* */
```

参数	含义
ref	允许此路径项的外部定义
response	状态
description	描述

Definition使用

Definition请求类型

在我们编写API文档时经常会遇到一些API的内容是重复的，现在我们就来学习如何抽取可以重新定义的内容，定义模型，来简化我们的文档。

示例1：

在index.php中的代码如下：

```
/**
* @SWG\Response(response="200",description="ok",@SWG\Schema(ref="#/definitions/index"))
*/
```

实现自定义definition

```
/**
 * @SWG\Definition(definition="index",example={"code"=100,"data"={},"msg"="string"})
 */
```

当然我们也可以通过定义一个类的方式来实现

示例2：

我们新建一个Definition.php

```
/**
 * @SWG\Definition()
 * @package app\index\controller
 */
class Definition{

    /**
     * @SWG\Property(example=0)
     */
    protected $code;
    /**
     * @SWG\Property(example={})
     */
    protected $data;

    /**
     * @SWG\Property(example="string")
     */
```

```
    protected $msg;
    public function index($result){

    }

}
```

Schema参数说明

Schema请求类型

参数 `@SWG\Schema()` 是输入和输出数据类型的定义，这些类型可以是对象，也可以是字符串和数组

```
/**
* @SWG\Post(
*     path="/index/index/user",
*     tags={"用户相关"},
*     summary="个人资料",
*     consumes={"application/json"},
*     produces={"application/json"},
*     @SWG\Parameter(
*     in="body",
*     required=true,
*     name="body",
*     description="description",
*     @SWG\Schema(example={"username"="username","password"="password"})
* )
*/
```

参数	含义
ref	允许此路径项的外部定义
example	设置一个自定义的对象
discriminator	添加多个自定义 `ref` 的支持

> 我们来看看参数 `@SWG\Schema()` 使用

> 使用 `example` 参数来自定义

```
/**
*     @SWG\Schema(example={"username"="username","password"="password"})
*/
```

> 使用 `discriminator` 参数来进行多个模型的支持

```
/**
*
*    @SWG\Response(response="200",description="ok",@SWG\Schema(ref="#/definitio
ns/Definition",discriminator="Definition")),
```

```
*    @SWG\Response(response="404",description="ok",@SWG\Schema(ref="#/definitio
ns/index",discriminator="index"))
*/
```

> 使用 `@SWG\Property()` 参数来设置：

参数	含义
property	名称
type	类型 可设置值为 string、 number、integer、boolean、array、object"
example	设置默认值
readOnly	属性声明

```
/** @SWG\Schema(
*     @SWG\Property(
*          property="username",
*          type="string",
*          example="username"
*     ),
*       @SWG\Property(
*          property="password",
*          type="string",
*          example="password"
*     ),
*     @SWG\Property(
*          property="sex",
*          type="integer",
*          example=0
*      )
*  )
*  /
```

> 参数 `readOnly` 如果设置为true，这意味着它只作为响应的一部分发送，不能作为请求的一部分发送，默认false

版本说明

版本说明

我们最后来聊聊最新版Swagger-php3.0的内容，最新版废除了一些参数，同时又新增了一些参数，我们以下面这段完整的post注释为例来说说：

```
/**
*
* @OA\SecurityScheme(type="apiKey",securityScheme="apikey",name="apikey")
* @OA\Post(
 *     tags={"教育模块"},
 *     path="/index/index/register",
 *     summary="注册接口",
 *     @OA\Parameter(name="firstname",in="query",@OA\Schema(type="integer"),des
cription="firstname"),
 *     security={{"apikey"={}}},
 *     @OA\RequestBody(required=true,description="body",content={
 *     @OA\MediaType(mediaType="application/json",
 *     @OA\Schema(
 *     @OA\Property(
 *        property="username",
 *        type="string"
 *     ),
 *     @OA\Property(
 *        property="password",
 *        type="string"
 *     ),
 *     @OA\Property(
 *        property="sex",
 *        type="integer",
 *        example=0
 *     ),
 * ))}
),
 *     @OA\Response(
 *     response=200,
 *     description="ok",
 * )
 * )
 */
```

> 属性 `in` 现在新增了值cookie，原先的值body，用 `@OA\RequestBody()` 代替，同时JSON的结构也有所变化， `@OA\SecurityScheme()` 参数type添加了值http、openIdConnect,删除了原来的 `basic`

想体验最新版的小伙伴可以去Swagger-PHP官网去查看最新的更新内容