**Swagger 开发规范**

# 简介

Swagger 是一款RESTFUL接口的文档在线自动生成+功能测试功能软件。

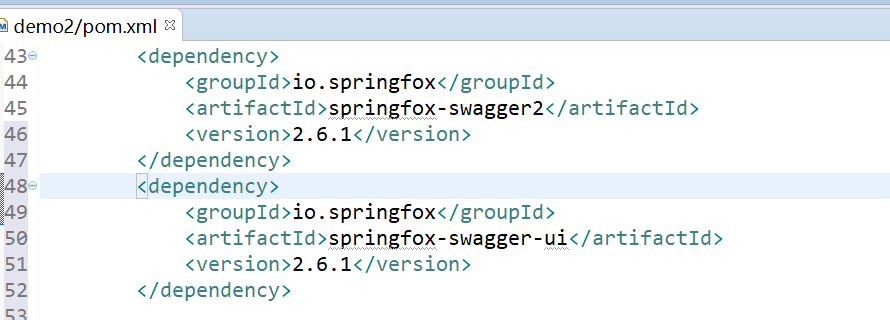
Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。总体目标是使客户端和文件系统作为服务器以同样的速度来更新。文件的方法，参数和模型紧密集成到服务器端的代码，允许API来始终保持同步。Swagger让部署管理和使用功能强大的API从未如此简单。

Swagger™的目标是为REST API定义一个标准的，与语言无关的接口，使人和计算机在看不到源码或者看不到文档或者不能通过网络流量检测的情况下能发现和理解各种服务的功能。当服务通过Swagger定义，消费者就能与远程的服务互动通过少量的实现逻辑。类似于低级编程接口，Swagger去掉了调用服务时的很多猜测。了解更多关于Swagger 项目的信息请访问：

https://github.com/OAI/OpenAPI-Specification

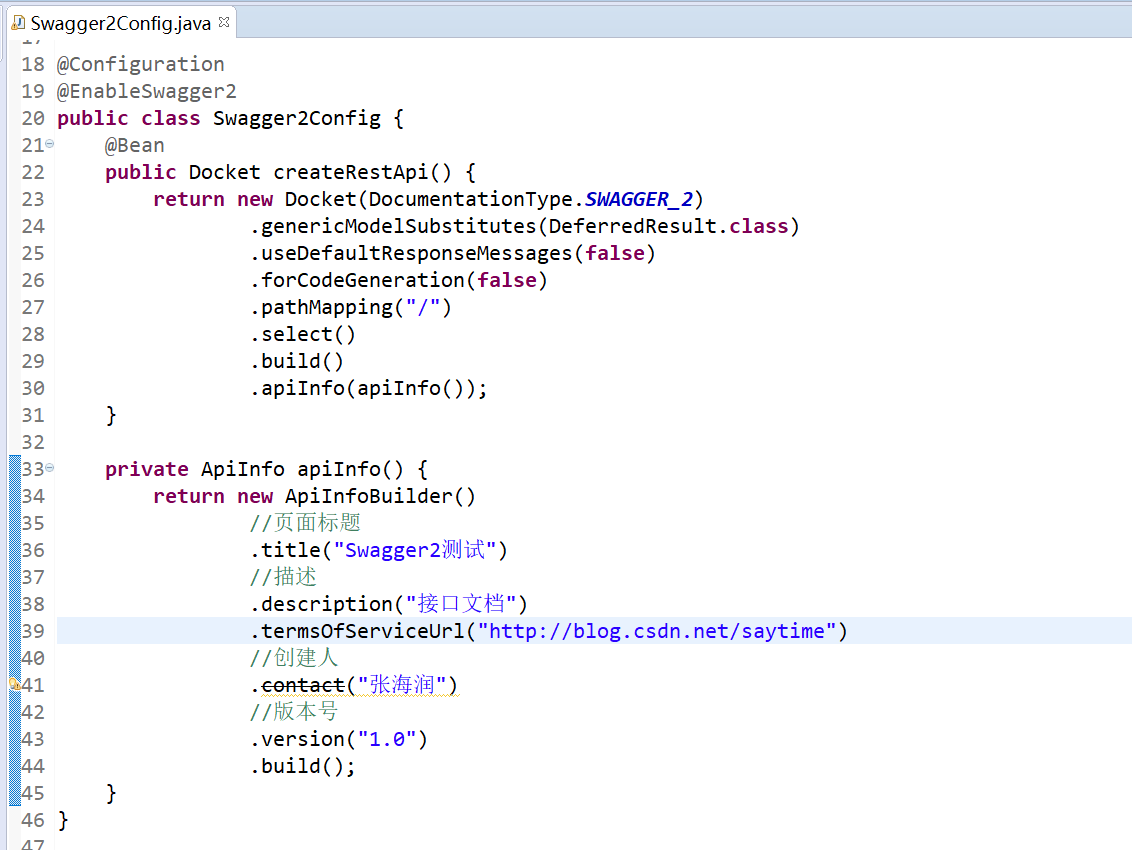
# SpringBoot+Swagger2框架整合

## 2.1依赖



## 2.2 Swagger配置类

Swagger配置类建议放置在包外层。特别要注意的是里面配置了api文件也就是controller包的路径，不然生成的文档扫描不到接口。



1.用@Configuration注解该类，等价于XML中配置beans；用@Bean标注方法等价于XML中配置bean。

2. .apis(RequestHandlerSelectors.basePackage("com.test.web"))配置controller包路径，便于Swagger扫描。

3.

.title()：文档标题

.description()：文档概要

.termsOfServiceUrl（）：（不可见）条款地址

.version（）：版本号

代码如下：

package com.example.demo;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.web.context.request.async.DeferredResult;

import springfox.documentation.builders.ApiInfoBuilder;

import springfox.documentation.builders.ParameterBuilder;

import springfox.documentation.schema.ModelRef;

import springfox.documentation.service.ApiInfo;

import springfox.documentation.service.Parameter;

import springfox.documentation.spi.DocumentationType;

import springfox.documentation.spring.web.plugins.Docket;

import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;

import java.util.List;

@Configuration

@EnableSwagger2

public class Swagger2Config {

@Bean

public Docket createRestApi() {

return new Docket(DocumentationType.SWAGGER\_2)

.genericModelSubstitutes(DeferredResult.class)

.useDefaultResponseMessages(false)

.forCodeGeneration(false)

.pathMapping("/")

.select()

.build()

.apiInfo(apiInfo());

}

private ApiInfo apiInfo() {

return new ApiInfoBuilder()

//页面标题

.title("Swagger2测试")

//描述

.description("接口文档")

.termsOfServiceUrl("http://blog.csdn.net/saytime")

//创建人

.contact("张海润")

//版本号

.version("1.0")

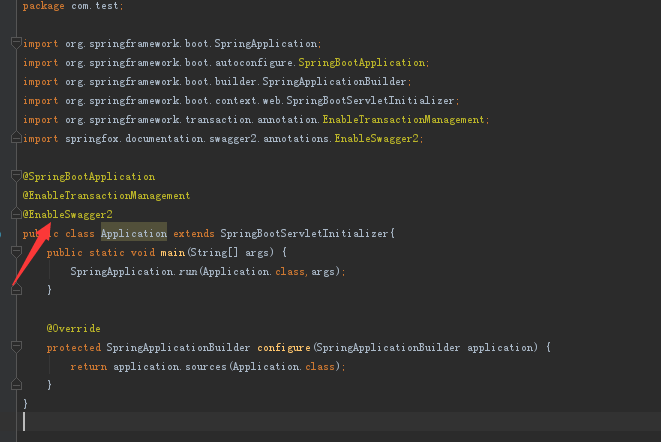
.build();

}

}

## 2.3 配置application.class

pplication.class 加上注解@EnableSwagger2 表示开启Swagger



# Swagger注解开发规范



## @Api

**注解说明：**

@Api 用在类上，说明该类的作用。可以标记一个Controller类做为swagger 文档资源。

**使用方式：**

@RestController  
@Api(value = **"user"**, tasgs= {**"获取用户信息接口"**})  
@RequestMapping(**"user"**)  
**public class** UserController {

}

**属性配置：**

|  |  |  |
| --- | --- | --- |
| **属性名称** | **备注** | **必填** |
| value | url的路径值 | 强制 |
| description | 对api资源的描述 | 强制 |
| tags | 如果设置这个值、value的值会被覆盖 | 建议 |
| basePath | 基本路径可以不配置 | 否 |
| position | 如果配置多个Api 想改变显示的顺序位置 | 否 |
| produces | For example, "application/json, application/xml" | 否 |
| consumes | For example,"application/json, application/xml" | 否 |
| protocols | Possible values: http, https, ws, wss. | 否 |
| Authorizations | 高级特性认证时配置 | 否 |
| hidden | 配置为true 将在文档中隐藏 | 否 |

## @ApiOperation

**注解说明：**

@ApiOperation用在方法上，说明方法的作用，每一个url资源的定义。与Controller中的方法并列使用。

**使用方式：**

@ApiOperation(value = **"/findUserList"**, notes = **"查询用户列表"**, httpMethod = **"POST"**)

**public** ResultDataDto findUserList(HttpServletRequest request) {  
 ResultDataDto rd = **new** ResultDataDto();  
 **return** rd;

}

**属性配置：**

|  |  |  |
| --- | --- | --- |
| **属性名称** | **备注** | **必填** |
| value | url的路径值 | 强制 |
| notes | 对api资源的描述 | 强制 |
| httpMethod | GET","HEAD","POST","PUT","DELETE","OPTIONS"and"PATCH" | 强制 |
| tags | 如果设置这个值、value的值会被覆盖 | 建议 |
| basePath | 基本路径可以不配置 | 否 |
| position | 如果配置多个Api 想改变显示的顺序位置 | 否 |
| produces | For example, "application/json, application/xml" | 否 |
| consumes | For example,"application/json, application/xml" | 否 |
| protocols | Possible values: http, https, ws, wss. | 否 |
| Authorizations | 高级特性认证时配置 | 否 |
| hidden | 配置为true 将在文档中隐藏 | 否 |
| response | 这些对象是有效的 "List", "Set" or "Map".，其他无效 | 否 |
| responseContainer | "GET", "HEAD", "POST", "PUT", "DELETE", "OPTIONS" and "PATCH" | 否 |
| code | http的状态码 默认 200 | 否 |
| extensions | 扩展属性 | 否 |

## @ApiResponses、@ApiResponse

**注解说明：**

@ApiResponse响应配置，用在@ApiResponses中，一般用于表达一个错误的响应信息。与Controller中的方法并列使用。

@ApiResponses响应集配置。与Controller中的方法并列使用。

**使用方式：**

@ApiResponse(code = 400, message = **"Invalid user"**,response = ResultDataDto.**class**)

**public** ResultDataDto findUserList(HttpServletRequest request) {  
 ResultDataDto rd = **new** ResultDataDto();  
 **return** rd;

}

或者

@ApiResponses

({

@ApiResponse(code = 400, message = **"Invalid user"**,response = ResultDataDto.**class**),

@ApiResponse(code = 200, message = **"SUCCESS"**,response = Viod.**class**)

})

**public** ResultDataDto findUserList(HttpServletRequest request) {  
 ResultDataDto rd = **new** ResultDataDto();  
 **return** rd;

}

**属性配置：**

|  |  |  |
| --- | --- | --- |
| **属性名称** | **备注** | **必填** |
| code | http的状态码 | 强制 |
| message | 描述 | 强制 |
| response | 默认响应类的Void | 建议 |
| reference | 参考ApiOperation中配置 | 否 |
| responseHeaders | 参考 ResponseHeader 属性配置说明 | 否 |
| responseContainer | 参考 ResponseHeader 属性配置说明 | 否 |

## @ApiModel、@ApiModelProperty、@ApiParam

**注解说明：**

@ApiModel描述返回对象的意义，用在对象类上。

@ApiModelProperty对象属性，用在出入参数对象的字段上。

@ApiParam 请求属性，用在请求参数上。

**使用方式：**

**类1**

@ApiModel(value = **"** UserSearchDto **"**,description =**"用户信息"**)  
**public class** UserSearchDto **implements** Serializable {

@ApiModelProperty(name = **"accountOption"**, value = **"账户状态 1禁用 2正常"**, dataType = **"long"**,requrired=**true** , example =**"1"**)

**private** Integer **accountOption**;

}

**类2**

**public** ResultDataDto findUserList(@RequestBody @ApiParam(name = **"查询用户列表"**, value = **"UserSearchDto类",** requrired=**true**) UserSearchDto userSearchDto, HttpServletRequest request) {  
 ResultDataDto rd = **new** ResultDataDto();  
 **return** rd;  
}

**属性配置：**

@ApiModel：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **备注** | **必填** |
| value | 为模型提供另一个名称 | 强制 |
| description | 对模型的描述 | 建议 |
| [discriminator](http://docs.swagger.io/swagger-core/apidocs/com/wordnik/swagger/annotations/ApiModel.html#discriminator()) | 支持类继承和多态性。 | 否 |
| parent | 提供模型的超类，以允许描述继承。 | 否 |
| subTypes | 从该模型继承的子类型的数组。 | 否 |

@ApiModelProperty：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **备注** | **必填** |
| value | 对该属性的简要描述 | 强制 |
| required | 是否必填 | 强制 |
| example | 举例说明 | 强制 |
| name | 重写属性名称 | 否 |
| Hidden | 配置为true 将在文档中隐藏 | 否 |

@ApiParam：

|  |  |  |
| --- | --- | --- |
| **属性名称** | **备注** | **必填** |
| name | 属性名称 | 强制 |
| required | 是否属性必填 | 强制 |
| value | 属性值 | 建议 |
| defaultValue | 默认属性值 | 否 |
| allowableValues | 可以不配置 | 否 |
| Access | 允许从API文档中过滤一个参数 | 否 |
| allowMultiple | 指定参数是否可以接受多个逗号分隔的值。默认为false | 否 |
| Hidden | 配置为true 将在文档中隐藏 | 否 |
| Example | 举例子 | 否 |

## @ApiImplicitParams、@ApiImplicitParam

**注解说明：**

@ApiImplicitParam用在@ApiImplicitParams注解中，指定一个请求参数的各个方面。

@ApiImplicitParams用在方法上包含一组参数说明。

**使用方式：**

@ApiImplicitParams

({  
 @ApiImplicitParam(name = **" userId "**, value = **"用户ID"**, dataType = **"long"**, paramType = **"query"**, requrired=**true** ,defaultValue=**" 999"**)  
 })

**public** ResultDataDto findUserList(Integer userId, HttpServletRequest request) {

ResultDataDto rd = **new** ResultDataDto();  
 **return** rd;  
 }

**属性配置：**

|  |  |  |
| --- | --- | --- |
| **属性名称** | **备注** | **必填** |
| name | 参数名 | 强制 |
| value | 描述 | 强制 |
| dataType | 参数数据类型 | 强制 |
| requrired | 是否是必填 | 强制 |
| paramType | 参数的参数类型 | 强制 |
| defaultValue | 描述参数的默认值 | 强制 |
| allowMultiple | 指定参数是否可以接受多个逗号分隔的值,默认为false | 否 |
| allowableValues | 限制此参数的可接受值 | 否 |
| access | 允许从API文档中过滤一个参数 | 否 |

详细注释api：

https://blog.csdn.net/xupeng874395012/article/details/68946676

源生注释api文档：https://github.com/wkennedy/swagger4spring-web

## API文档展示

完成上述代码添加上，启动Spring Boot程序，访问：<http://ip:port/swagger-ui.html>，就能看到前文所展示的API的页面。我们可以再点开具体的API请求，以POST类型的/users请求为例，可找到上述代码中我们配置的Notes信息以及参数user的描述信息。

## 效果图

