> 学习自https://www.jianshu.com/p/dcaff30f91cc?tdsourcetag=s_pctim_aiomsg
> 以及https://github.com/swagger-api/swagger-core/wiki/Annotations-1.5.X
> https://blog.csdn.net/java_yes/article/details/79183804

OpenAPI

OpenAPI规范(以前称为Swagger规范)是REST APi的API描述格式。OpenAPI允许扫描整个APi。包括：

1.每个端点上的可用端点(/usres)和操作(GET/usres,POST/users)

2.操作参数的每个输入和输出

3.认证方法

4.联系信息，许可证，使用条款和其他信息

5.APi规范可以用YAML或JSON编写。

Swagger

Swagger是一套围绕OpenAPI规范构建的开源工具，它可以帮助我们设计、构建、编写和使用RestAPI。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。

作用：

接口文档在线自动生成

功能测试

主要工具

Swagger编辑器：基于浏览器的编辑器，我们可以在其中编写Open

API规范：Swagger UI将API规范呈现为交互式API文档。

Swagger Codegen：根据　　OpenAPI规范生成服务器代码和客户端库。

Swagger是一组开源项目，其中主要项目如下：

1.Swagger-tools:提供各种与Swagger进行集成和交互的工具。

2.Swagger-core:用于Java/Scala的Swagger实现。

3.Swagger-js: 用于JavaScript的Swagger实现。

4.Swagger-node-express:Swagger模块，用于node.js的Express web应用框架。

5.Swagger-ui:一个无依赖的HTML、JS和CSS集合，可以为Swagger兼容API动态生成优雅文档。

Swagger-codegen:一个模板驱动引擎，通过分析用户Swagger资源声明以各种语言生成客户端代码。

Swagger使用的注解及其说明

> **具体详情**http://docs.swagger.io/swagger-core/v1.5.X/apidocs/index.html?io/swagger/annotations/ApiOperation.html

@Api

作用在类上，用来标注该类具体实现内容。将类标记为Swagger资源。

参数：

value:默认值。

tags：允许为操作设置多个标记的属性，不是使用。

@ApiImplicitParam

作用在方法上，表示单独的API操作参数。

参数：

value:参数的具体意义作用。

required:是否必填。

name:参数名。

dataType：参数的数据类型。

paramType：查询参数类型。

@ApiImplicitParams

用于方法，包含多个@ApiImplicitParam

```
@ApiImplicitParams（{
  @ApiImplicitParam（name = “ name ”, value = “ User's name ”, required = true, dataType = “ string ”, paramType = “ query ”）,
  @ ApiImplicitParam（name = “ email ”, value = “ User's电子邮件“, required = false, dataType = “ string ”, paramType = “ query ”）,
  @ ApiImplicitParam（name = “ id ”, value = “ User ID ”, required = true, dataType = “ long ”, paramType = “ query ”）
  }）
  public void doPost（HttpServletRequest request, HttpServletResponse response）抛出ServletException, IOException { ... }
```

@ApiOperation

用于方法，表示一个http请求操作。

```
@GET
  @Path("/findByStatus")
  @ApiOperation(value = "Finds Pets by status",
  notes = "Multiple status values can be provided with comma seperated strings",
  response = Pet.class,
  responseContainer = "List")
  public Response findPetsByStatus(...) { ... }
```

@ApiModel

用于类，表示对类进行说。

用于参数，用实体类接收。

@ApiModelProperty

用于方法字段，表示对model属性的说明或者数据操作更改。

```
@ApiModel(value = "User", description = "用户")
public class User implements Serializable{
 /**
 * 用户ID
 */
 @ApiModelProperty(value = "用户ID", required = true)
 @NotEmpty(message = "{id.empty}", groups = {Default.class,New.class,Update.class})
 protected String id;
}
```

@ApiResponse

用于方法，描述操作的可能响应。

@ApiResponses

用于方法，一个允许多个@ApiResponse对象列表的包装器。

```
 @ApiResponses(value = {
 @ApiResponse(code = 400, message = "Invalid ID supplied",
 responseHeaders = @ResponseHeader(name = "X-Rack-Cache", description = "Explains whether or not a cache was used", response = Boolean.class)),
 @ApiResponse(code = 404, message = "Pet not found") })
 public Response getPetById(...) {...}
```

@ApiParam

用于方法、参数、字段说明，表示对参数的添加元数据

@Authorization

声明要在资源或操作上使用的　授权方案

@AuthorizationScope:

介绍一个OAuth2授权范围。

@ResponseHeader

响应头设置，使用方法。

Spring Boot集成Swagger

```
<!--Swagger依赖-->
<dependency>
 <groupId>io.springfox</groupId>
 <artifactId>springfox-swagger2</artifactId>
 <version>2.2.2</version>
</dependency>
<dependency>
 <groupId>io.springfox</groupId>
```

```
 <artifactId>springfox-swagger-ui</artifactId>
  <version>2.2.2</version>
</dependency>
```

配置Swagger

在同级别的Application.java中创建Swagger2.java

```
@Configuration
@EnableSwagger2
public class Swagger2 {

 @Bean
 public Docket createRestApi(){
 return new Docket(DocumentationType.SWAGGER_2)
 .apiInfo(apiInfo())
 .select()
  .apis(RequestHandlerSelectors.basePackage("com.yi.feng.helloswagger.con
troller"))
  .paths(PathSelectors.any())
  .build();
  }

  /**
  * 创建Api基本信息（这些会展示在文档页面）
  * @return
  */
  private ApiInfo apiInfo(){
  return new ApiInfoBuilder()
  .title("SpringBoot中使用Swagg2构建Restful Api")
  .description("更多详情 请听下回分解")
  .termsOfServiceUrl("https://github.com/peakbro/HelloGit")
  .contact("峰")
  .version("1.0")
  .build();
  }
}
```

添加UserController的index方法

```
@RestController
@Api(value = "欢迎使用Swagger2",tags = "欢迎使用Swagger2")
public class HelloController {
 @RequestMapping(value = "/hello",method = RequestMethod.GET)
 public String index(){
```

```
6   return "Hello World";
7   }
8  }
```

效果如下