链滴

swagger 介绍以及 spring 自动化整合

作者：shixiaoxiang

原文链接：https://ld246.com/article/1508032820734

来源网站：链滴

Swagger 简单介绍
<blockquote>
Swagger 是一种 Rest API 的表示方式，它与语言无关，不但人可读，而且机器可读。简单来说 Swagger 可以用作 Rest API 的交互式文档。
</blockquote>
本文将从 Swagger 的一下组件介绍 swagger:是易于学习和可读的。

Swagger API Spec，描述 Rest API 的语言

Swagger UI，将 Swagger API Spec 以 HTML 页面展现出来

Swagger Editor，Swagger API Spec 的编辑器

Swagger Codegen ，从 OpenAPI 规范生成服务器存根和客户端库。

Swagger API Spec/Open API Spec
OpenAPI 规范（以前的 Swagger 规范）是 REST API 的 API 描述格式。OpenAPI 文件允许您述整个 API，包括：

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

操作参数每个操作的输入和输出

认证方式

联系信息，许可证，使用条款和其他信息。
API 规范可以用 YAML 或 JSON 编写。这种格式对于人类和机器都是易于学习和可读的。
完整的 OpenAPI 规范可以在 GitHub：
OpenAPI 3.0 规范中找到
关于 Swagger API Spec 包含的内容，可以查看官网文

Swagger UI
Swagger UI 是 Swagger 中用于显示 Rest 接口文档的项目，项目由一组 HTML，JavaScript 和 CSS 组成，没有外部依赖。Swagger UI 可以根据 Swagger Spec 的 json 动态生成漂亮的帮助文档支持常见浏览器。
可以访问官方在线 Swagger UI：http://petstore.swagger.io/<a>
使用 Swagger UI，可以将项目下载到本地 <a href="https://ld246.com/forward?goto=http%3A%2F%2Fgithub.com%2Fswagger-api%2Fswagger-ui" target="_blank" rel="nofollow ugc"

https://github.com/swagger-api/swagger-ui
然后使用浏览器打开 dist/index.html 就行了，可以将项目放到 HTTP Server 中通过 HTTP 协
访问。
将 index.html 页面的 json 文件修改成我们自己的 API 描述文件就行了。
Swagger Editor
Swagger Editor 是 Swagger API Spec 的编辑器，Swagger API Spec 有 2 中格式，yaml 和 js
n，Swagger Editor 使用 yaml 进行编辑，但允许导入和下载两种格式的文件。在 yaml 编辑器的右
有所见即所得的预览。
Swagger Editor 的官方在线：<a href="https://ld246.com/forward?goto=http%3A%2F%2Fl
nk.zhihu.com%2F%3Ftarget%3Dhttp%253A%2F%2Feditor.swagger.io%2F%2523%2F" target=
_blank" rel="nofollow ugc">Swagger Editor
Swagger Editor 的安装也很方便，下载最新的发布版：<a href="https://ld246.com/forward?
oto=https%3A%2F%2Fgithub.com%2Fswagger-api%2Fswagger-editor" target="_blank" rel="
ofollow ugc">https://github.com/swagger-api/swagger-editor,然后解压到文件夹，用 HT
P Server 将静态文件加载起来，下面是安装 node.jsHTTP Server 并跑起来的指令
npm install -g http-server
wget <a href="https://ld246.com/forward?goto=https%3A%2F%2Fgithub.com%2Fswag
er-api%2Fswagger-editor%2Freleases%2Fdownload%2Fv2.10.1%2Fswagger-editor.zip" targe
="_blank" rel="nofollow ugc">https://github.com/swagger-api/swagger-editor/releases/dow
load/v2.10.1/swagger-editor.zip
unzip swagger-editor.zip
http-server -p 8080 swagger-editor
HTTP Server 启动后，就可以在浏览器中输入地址进行编辑了。
文件菜单提供了主要的功能

New，创建新的文件
Open Example，打开内建 Swagger API Spec 的示例
Paste Json，将剪贴板的内容贴到编辑器中，取代当前的内容。在 Paste 之前一定要先下载编辑
的内容
Import URL/Import File，导入已有的 Swagger API Spec，可以是 yaml 或 json 格式的
Download YAML/Download JSON，将编辑的结果下载到本地。

Swagger Codegen
未进行深入学习
Swagger 与 Spring 进行整合
在 Spring 项目中我们可以使用 Swagger 帮我们生成 Rest API 的文档，一般有两种方式。

在 Resource 中引入 Swagger UI，通过自己编写 Swagger API 的 json 文件，将 index.html
的 json 文件进行替换的方式进行整合

通过引入依赖 springfox 来完成整合，这种整合方式需要我们在 api 的接口上添加注解，使代码
起来比较冗余，但是这种方式避免了编写繁琐的 yml 文件，可以自动帮我们生成并引入 Swagger UI
中
下面介绍一下第二种方式：

1、添加依赖

```
io.springfox
springfox-swagger2
2.6.1
```

2、添加 swagger 配置文件
<pre><code class="highlight-chroma">@Configuration

@EnableSwagger

public class SwaggerConfig {

    @Bean
    public Docket api() {
        return new Docket(DocumentationType.SWAGGER_2)
                .select()
// 选择那些路径和api会生成document
                .paths(Predicates.not(PathSelectors.regex("/error.*")))
                // .apis(RequestHandlerSelectors.basePackage("com.taikang.im.login.controller")) // 根据包名选择
                .apis(RequestHandlerSelectors.any()) // 对所有api进行监控
                .paths(PathSelectors.any()) // 对所有路径进行监控
                .build();
    }

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .title("Sping Boot中使用Swagger2构建RESTful APIs")
                .description("随便写点")
                .termsOfServiceUrl("随便写点")
                .contact("程序猿DD")
                .version("1.0")
                .build();
    }

}
</code></pre>
通过注解 EnableSwagger2 声明 Swagger 的可用性，此处会定义一个类型为 Docket 的 bean
关于 docket 类的说明如下：
<blockquote>
A builder which is intended to be the primary interface into the swagger-springmvc framework.Provides sensible defaults and convenience methods for configuration.
</blockquote>
Docket 的 select()方法会提供给 swagger-springmvc framework 的一个默认构造器（ApiSele

torBuilder），这个构造器为配置 swagger 提供了一系列的默认属性和便利方法。例如可以控制哪接口暴露给 Swagger 来展现，本例采用指定扫描的包路径来定义，Swagger 会扫描该包下所有 Controller 定义的 API，并产生文档内容（除了被 @ApiIgnore 指定的请求）。
<code>apiInfo()</code> 用来创建该 Api 的基本信息（这些基本信息会展现在文档页面中）。

3、添加文档内容
<pre><code class="highlight-chroma">1. @ApiOperation(value="测试日志程序", notes="测试日志程序", produces = "application/json")
2.   @ApiImplicitParams(value = {
3.   @ApiImplicitParam(name = "school", value = "学校名称", required = true, paramType = "query", dataType = "String"),
4.   @ApiImplicitParam(name = "name", value = "姓名", required = true, paramType = "query", dataType = "String")
5.   })
6.   @RequestMapping(value = "/get",method = RequestMethod.GET)
7.   public void testProgram(HttpServletRequest request, HttpServletResponse response){
8.   logger.debug("debug");
9.   logger.info("info");
10.   logger.error("error");
11.   logger.warn("warn");
12.
13.   Student student = studentCommandService.selectById(1);
14. //   Student student = studentCommandService.queryById(1l);
15.   System.out.println(student.toString());
16.   this.renderJson(CodeEnum.success.getValue(),CodeEnum.success.getDescription(),student,request,response);
17.   }
</code></pre>
注解说明：
@Api：用在类上，说明该类的作用
@ApiOperation：用在方法上，说明方法的作用
@ApiImplicitParams：用在方法上包含一组参数说明
@ApiImplicitParam：用在 @ApiImplicitParams 注解中，指定一个请求参数的各个方面
paramType：参数放在哪个地方
header--> 请求参数的获取：@RequestHeader
query--> 请求参数的获取：@RequestParam
path（用于 restful 接口）--> 请求参数的获取：@PathVariable
body（不常用）
form（不常用）
name：参数名

dataType：参数类型
required：参数是否必须传
value：参数的意思
defaultValue：参数的默认值
@ApiResponses：用于表示一组响应
@ApiResponse：用在 @ApiResponses 中，一般用于表达一个错误的响应信息
code：数字，例如 400
message：信息，例如"请求参数没填好"
response：抛出异常的类
@ApiModel：描述一个 Model 的信息（这种一般用在 post 创建的时候，使用 @RequestBody 这的场景，请求参数无法使用 @ApiImplicitParam 注解进行描述的时候）
@ApiModelProperty：描述一个 model 的属性
<hr>
swagger-ui 测试 rest 接口
1、添加依赖
<pre><code class="highlight-chroma"><dependency>
    <groupId>
o.springfox</groupId>
    <artifactId&g
;springfox-swagger-ui</artifactId>
    <version>
.4.0</version>
</dependency
gt;
</code></pre>
通过访问：http://localhost:8080/your-app-root/v2/api-docs，能测试生成的 api 是否可用此时返回的是一个 json 形式的页面，可读性不好。可以通过 Swagger UI 来生成一个可读性良好的 a i 页面。
再次访问：http://localhost:8080/your-app-root/swagger-ui.html 就可以看到可读性较好的 pi 文档页面。
API Security
有时候我们需要在 API 上添加认证，在 Swagger 规范中定义了一些认证的方式以及 yml 编写的规范，需要注意的是，<a href "https://ld246.com/forward?goto=https%3A%2F%2Fswagger.io%2Fdocs%2Fspecification%2F -0%2Fauthentication%2F" target="_blank" rel="nofollow ugc">Swagger2.0 和 Swagger3.0 中关于认证的方式略有同，Swagger3.0 中多了一个承载认证(Bearer Authentication)，即 token 认证，而在 Swagger2 0 规范中没有，所以我们可以使用 API 密钥认证(API Keys)进行替代，关于两种规范之间的不同，可以自行查阅 方文档
具体配置参考 Springfox Spring MVC 和 Spring Boot
值得一提的是最新版本的 Springfox 实现的规范是 Swagger2.0 而不是 Swagger3.0
注解部分：
具体参考 https://github.com/swagger-api/swagger-core/wiki/Annotati

ns-1.5.X#quick-annotation-overview
<h3 id="-Api">@Api</h3>
In Swagger 2.0, resources were replaced by tags, and this impacts the @Api annotation. It is no longer used to declare a resource, and it is now used to apply definitions for all the operations defined under it.
A JAX-RS usage would be:
<pre><code class="highlight-chroma">@Path("/pet")

@Api(value = "pet", authorizations = {

@Authorization(value="sampleoauth", scopes = {})

})

@Produces({"application/json", "application/xml"})

public class PetResource {

...

}
</code></pre>
In this example, we're saying that the tag for the operations under this class is pet (so the would all be grouped together). Swagger will pick up on the @Produces annotation but you an override this value if you wish.
@Api can also be used to declare authorization at the resource-level. These definitions aply to all operations under this resource, but can be overridden at the operation level if needd. In the example above, we're adding a previously-declared OAuth2 authorization scheme without any scopes. For further details, check the @Authorizationnnotation.
Instead of using the value(), you can use the tags() property which allows you to set multple tags for the operations. For example:
@Api(tags = {"external_info","user_info"})
Note that in this case, value() would be ignored even if it exists.
The boolean hidden property can be used to entirely hide an @Api even if it declared. Ths is especially useful when using sub-resources to remove unwanted artifacts.
In swagger-core 1.5.X, description(), basePath(), and position() are no longer used.
For further details about this annotation, usage and edge cases, check out the javadocs.
<h3 id="-ApiResponses---ApiResponse">@ApiResponses, <a href="https://ld246.com/forward?goto=http%3A%2F%2Fdocs.swagger.io%2Fs

agger-core%2Fcurrent%2Fapidocs%2Findex.html%3Fio%2Fswagger%2Fannotations%2FApiResponse.html" target="_blank" rel="nofollow ugc">@ApiResponse</h3>
It's a common practice to return errors (or other success messages) using HTTP status codes. While the general return type of an operation is defined in the @ApiOperation, the rest of the return codes should be described using these annotations.
The @ApiResponse describes a concrete possible response. It cannot be used directly on the method or class/interface and needs to be included in the array value of @ApiResponses whether there's one response or more).
If the response is accompanied with a body, the body model can be described as well (one model per response).
<pre><code class="highlight-chroma">@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 etPetById(...) {...}
</code></pre>
In swagger-core 1.5.X, you can also add description of response headers as seen in the example above.
For further details about this annotation, usage and edge cases, check out the javadocs (@ApiResponses, @ApiResponse).
遇到的问题
我们在使用 SpringBoot 集成 Swagger2 中，访问：http://127.0.0.1:8080/swagger-ui.html
<h3 id="问题描述">问题描述</h3>
可能出现两种错误：
1.页面显示默认报错页面。后台报错：
No handler found for GET /swagger-ui.html
2.显示 Swagger 空白页面：
后台报错：
No mapping found for HTTP request with URI [/swagger-resources/configuration/ui] in DispatcherServlet with name 'dispatcherServlet'
<h3 id="解决方案">解决方案</h3>
这个错误，是因为资源映射问题导致。
我们在访问 http://127.0.0.1:8188/swagger-ui.html 时，这个 swagger-ui.html 相关的所有前端静态文件都在 springfox-swagger-ui-2.6.1.jar

面。目录如下：
Spring Boot 自动配置本身不会自动把/swagger-ui.html 这个路径映射到对应的目录 META-INF resources/下面。我们加上这个映射即可。代码如下：
<pre><code class="highlight-chroma">@Configuration

class WebMvcCon ig extends WebMvcConfigurerAdapter {

    @Override

    void addResour eHandlers(ResourceHandlerRegistry registry) {

        registry.addR sourceHandler("swagger-ui.html")

                .addRes urceLocations("classpath:/META-INF/resources/")

        registry.addR sourceHandler("/webjars/**")

                .addRes urceLocations("classpath:/META-INF/resources/webjars/")

    }
</code></pre>