爱旅行项目—使用 Swagger 自动生成 API 说明文档

随之 API 的发展以及需求的日益增加，对 API 的说明文档需求也随之而来，特别是对于前后端分离的项目，前后端团队协作开发，前端开发人员希望能通过标准、清晰、完整的接口描述文档来正确的进行后端接口的调用，从而完成前后数据的交互，并有效地减少前后端团队之间的沟通成本。作为后端开发人员，只需要接口测试通过即可，确保数据的准确性，不必关心产品，只提供数据。当然 API 描述文档若通过手写的方式编写，将会产生巨大的工作量，那么在爱旅行项目中，我们使用 Swagger 自动生成 API 文档。

1.1 Swagger 简介

Swagger 是一个简单、强大的 Restful API 文档生成管理工具，通过 swagger-spring 项目实现了与 Sping MVC 框架的无缝集成功能，方便生成 spring restful 风格的接口文档，在项目中集成这个工具，根据我们自己的配置信息能够自动为我们生成一个 API 文档展示页，可以在浏览器中直接访问查看项目的接口信息(如下图 1 所示)，同时 swagger-ui 还可以测试 spring restful 风格的接口功能，可以对项目提供的每一个 API 接口进行相应的测试。Swagger 生成的 API 文档是实时更新的，API 接口有任何改动都会在文档中及时的表现出来。其官方网站为：http://swagger.io/，下面我将详细介绍在爱旅行项目中如何集成 Swagger。

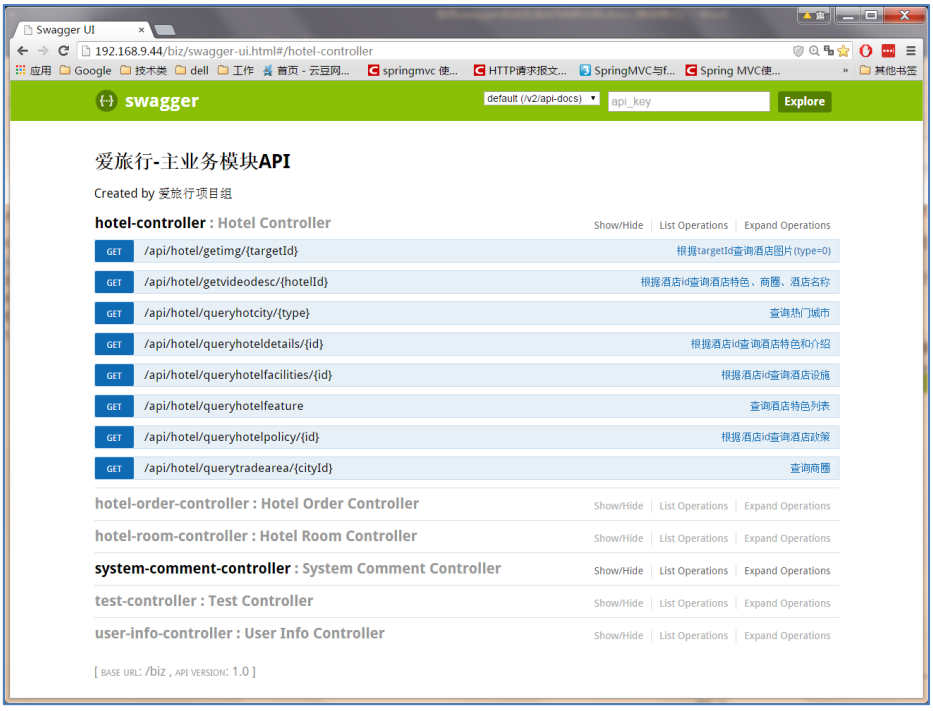

图 1

1.2 项目集成 Swagger

1.2.1 项目环境

Spring 提供了一个与 Swagger 的集成工具包 springfox，让 Spring 项目能够更好的与 Swagger 融合，官网提供两个版本可完成集成：

- swagger-springmvc
- springfox-swagger2

两种版本配置不同，爱旅行项目中采用新版本 springfox-swagger2。项目运行环境：

- JDK1.8（注：必须使用 JDK1.8 否则 swagger2 无法运行）
- Spring 4.1.7
- Mybatis3.2.2

项目中所需 Maven 依赖：

- springfox-swagger2
- springfox-swagger-ui
- guava
- mapstruct-jdk8
- Jackson
  - ✓ jackson-core
  - ✓ jackson-databind
  - ✓ jackson-annotations

1.2.2 配置步骤

1. 在 pom.xml 文件中添加 Swagger2 相关的依赖，配置（部分）如下：

```
<!—Swagger api 文档生成工具依赖包-->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.4.0</version>
```

```
    <exclusions>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
        </exclusion>
        <exclusion>
            <groupId>com.fasterxml</groupId>
            <artifactId>classmate</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.4.0</version>
    <exclusions>
        <exclusion>
            <groupId>org.springframework</groupId>
            <artifactId>spring-aop</artifactId>
        </exclusion>
    </exclusions>
</dependency>
<dependency>
    <groupId>com.google.guava</groupId>
    <artifactId>guava</artifactId>
    <version>19.0</version>
</dependency>
<dependency>
    <groupId>org.mapstruct</groupId>
```

```
        <artifactId>mapstruct-jdk8</artifactId>
        <version>1.1.0.Final</version>
</dependency>
<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-core</artifactId>
        <version>${jackson.verson}</version>
</dependency>
<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-databind</artifactId>
        <version>${jackson.verson}</version>
</dependency>
<dependency>
        <groupId>com.fasterxml.jackson.core</groupId>
        <artifactId>jackson-annotations</artifactId>
        <version>${jackson.verson}</version>
</dependency>
```

2. Swagger2 配置类：SwaggerConfig.java（可官网下载）

```java
@EnableSwagger2
@ComponentScan(basePackages = {"cn.itrip.controller"})
@Configuration
public class SwaggerConfig extends WebMvcConfigurationSupport {
    @Bean
    public Docket createRestApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                .apis(RequestHandlerSelectors.any())
                .paths(PathSelectors.any())
                .build();
    }
    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .title("爱旅行-主业务模块API")
                .termsOfServiceUrl("http://itrip.project.bdqn.cn/biz")
                .contact("爱旅行项目组")
                .version("1.0")
                .build();
    }
}
```

图 2

1> @ComponentScan(basePackages = {"cn.itrip.controller"}):设置 Swagger 扫描包:

2> @EnableSwagger2: 使 Swagger2 生效

3> @Configuration:自动在本类上下文加载一些环境变量信息

4> 配置 ApiInfoBuilder(界面显示)

3. Spring MVC 配置文件

1> <mvc:default-servlet-handler />:配置对静态文件的处理方式

```
<!-- 使用 Swagger Restful API 文档时，添加此注解 -->
<mvc:default-servlet-handler />
```

2> <context:component-scan/>:添加指定扫描

```
<context:component-scan base-package="cn.itrip.controller">
   <context:include-filter type="annotation" expression="org.springframework.stereotype.Controller"/>
   <context:include-filter type="annotation" expression="org.springframework.context.annotation.Configuration"/>
   <context:include-filter type="annotation" expression="org.springframework.scheduling.annotation.Scheduled"/>
</context:component-scan>
```

1.2.3 Nginx 配置

Swagger 项目集成配置完成之后，为了能够实现对 Swagger 界面的访问，我们还需要修改 Nginx 的配置文件（nginx.conf）。注：爱旅行项目通过 Nginx 配置的 upstream 反向代理到后端服务，并且为了安全考虑，服务器的端口只开放 80 端口，Tomcat 端口均不开放。具体的配置（nginx.conf）修改如下图 3 所示：

```
server {
    listen       80;
    server_name  itrip.project.bdqn.cn;
    #root /data/itrip/itripfront; #前端静态工程
    index index.html;

    location /biz {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://itripbiz_server;
        }
    location /search {
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_pass http://itripsearch_server;
        }
```

中间部分配置省略...

```
#生产环境放开以下代码，同时不能访问swagger
#location ~ .*\.(gif|jpg|jpeg|png|bmp|swf|ioc|rar|zip|txt|flv|mid|doc|ppt|pdf|xls|mp3|wma|
html|woff|ttf|mp4|webm)$
#   {
#   expires 6h;
#   }

#location ~ .*\.(js|css)?$
#   {
#    expires 2h;
#   }
```

图 3

红框部分的配置均需要注释掉，才能实现 Swagger 的访问，否则放开注释将会往 root 指定的/data/itrip/itripfront 里请求，即访问前端页面。

通过 Nginx 正常访问 Swagger 界面：http://IP:port/{context-path}/swagger-ui.html

e.g:

访问 itripbiz 模块的 Swagger 界面：http://192.168.9.44/biz/swagger-ui.html

1.2.4 具体运用

Swagger 项目集成完成之后，具体的运用就是在 API 中加入 Swagger，实际上在原有的 SpringMVC 系统中（项目中）添加 Swagger 很简单，就是通过在接口上添加注解实现，接口

文档的同步效果。以 SystemCommentController.java 为例：

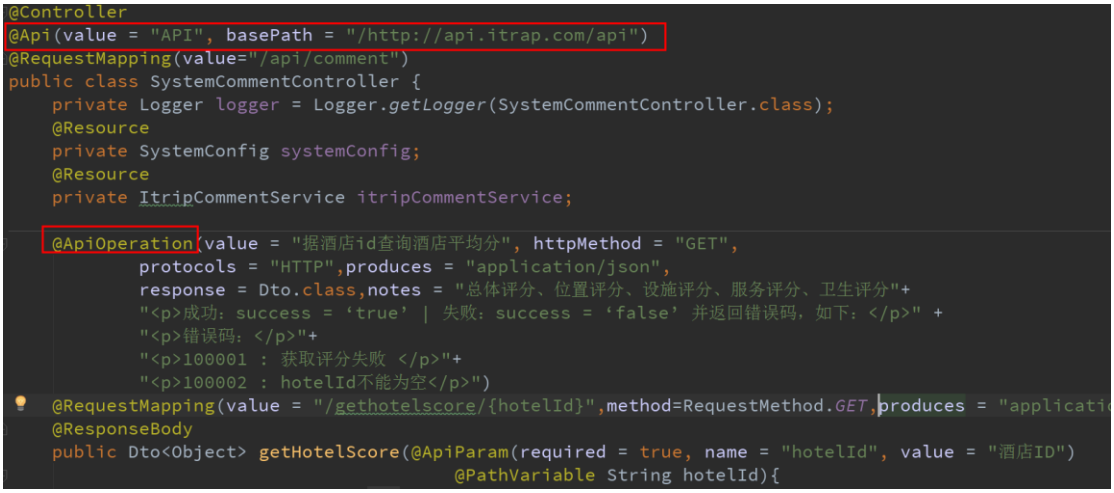

图 4

- @Api：用在类上面，说明该类的作用，即：该类为可执行测试的开放 API。
- @ApiOperation：用在方法上面，value=“接口说明”,httpMethod = “接口请求方式”, response = “接口返回的对象”, notes = “接口发布说明” produces 用来标记 API 返回值的具体类型，Protocols：通讯协议（http, https 等）。
- @ApiParam：用在方法参数前面，单个参数描述。

运行 swagger-ui 测试接口，访问 Swagger 界面（http://192.168.9.44/biz/swagger-ui.html#/），该接口描述如下图 5 所示：

图 5

输入参数 hotelId，点击【Try it out!】测试接口，查看返回数据，测试结果如下图 6 所示：

Request URL

```
http://192.168.9.44/biz/api/comment/gethotelscore/1
```

Request Headers

```
{
  "Accept": "application/json"
}
```

Response Body

```
{
  "data": {
    "avgFacilitiesScore": 4,
    "avgHygieneScore": 4.1,
    "avgPositionScore": 4.1,
    "avgScore": 4.4,
    "avgServiceScore": 2.9
  },
  "errorCode": "0",
  "msg": "获取评分成功",
  "success": "true"
}
```

Response Code

```
200
```

Response Headers

```
{
  "date": "Mon, 03 Jul 2017 11:51:47 GMT",
  "via": "1.1 VSERVER",
  "proxy-connection": "Keep-Alive",
  "connection": "Keep-Alive",
  "content-length": "177",
  "server": "Tengine",
  "content-type": "application/json;charset=UTF-8"
}
```

图 6

通过上述示例，我们发现该 API 方法描述非常清晰，包括接口描述，参数类型，返回的数据类型以及错误码、200 的状态码等等。但是本例中参数较为简单，若入参为一个对象时，接口描述就需要对参数的各个字段进行详细的描述，比如：须告之前端，参数字段是否为必填项，参数字段的接收类型等等。比如下面的这个接口（根据评论类型查询评论列表，并分页显示）如下图 7 所示：

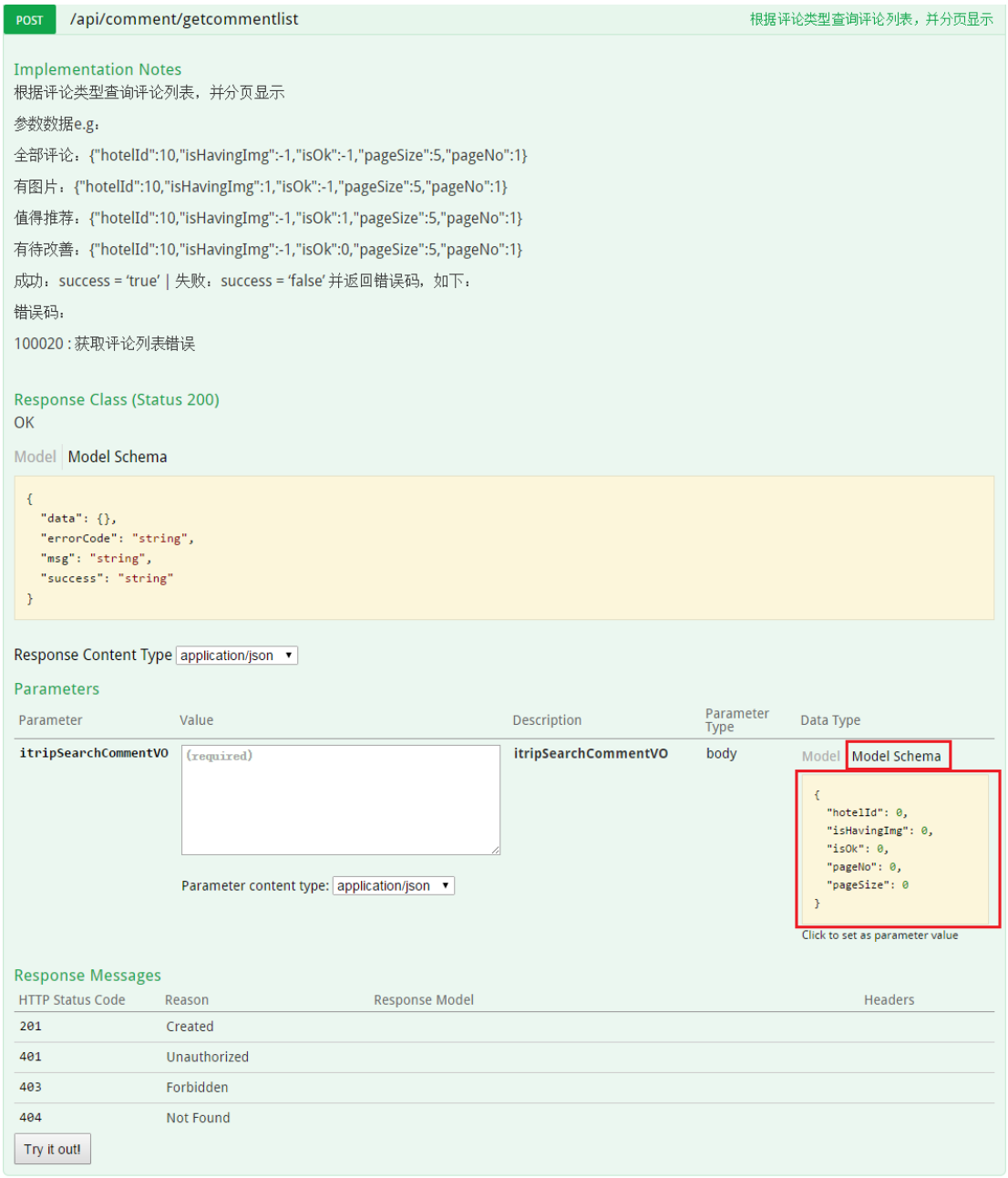

图 7

在上图中入参为对象，Model Schema 中描述了所需的参数字段，但是前端人员并不能很好理解各个属性字段所代表的意思以及用户输入时的一些限定，比如：必要的字段需要增加相应的 JS 验证等，那么可通过 Model 进行相应的属性定义描述，修改参数对象（itripSearchCommentVO.java）,如下图 8 所示：

```java
/**
 * 前端输入-评论搜索条件VO
 * Created by hanlu on 2017/5/10.
 */
@ApiModel(value = "ItripSearchCommentVO",description = "搜索评论VO")
public class ItripSearchCommentVO {

    @ApiModelProperty("[必填] 酒店ID")
    private Long hotelId;

    @ApiModelProperty("[必填，注：接收数字类型] 是否有评论照片（0 无图片 1 有图片）")
    private Integer isHavingImg;//是否有评论图片（0 无图片 1 有图片）

    @ApiModelProperty("[必填，注：接收数字类型] 是否满意（0：有待改善 1：值得推荐）")
    private Integer isOk;//是否满意（0：有待改善 1：值得推荐）

    @ApiModelProperty("[必填] 页面容量")
    private Integer pageSize;

    @ApiModelProperty("[必填] 页码）")
    private Integer pageNo;
```

图 8

- @ApiModel：当用对象来接收参数时，描述一个 Model 的信息
- @ApiModelProperty：描述 Model 的属性

运行 swagger-ui 测试接口（http://192.168.9.44/biz/swagger-ui.html#/），该接口描述如下图 9 所示：

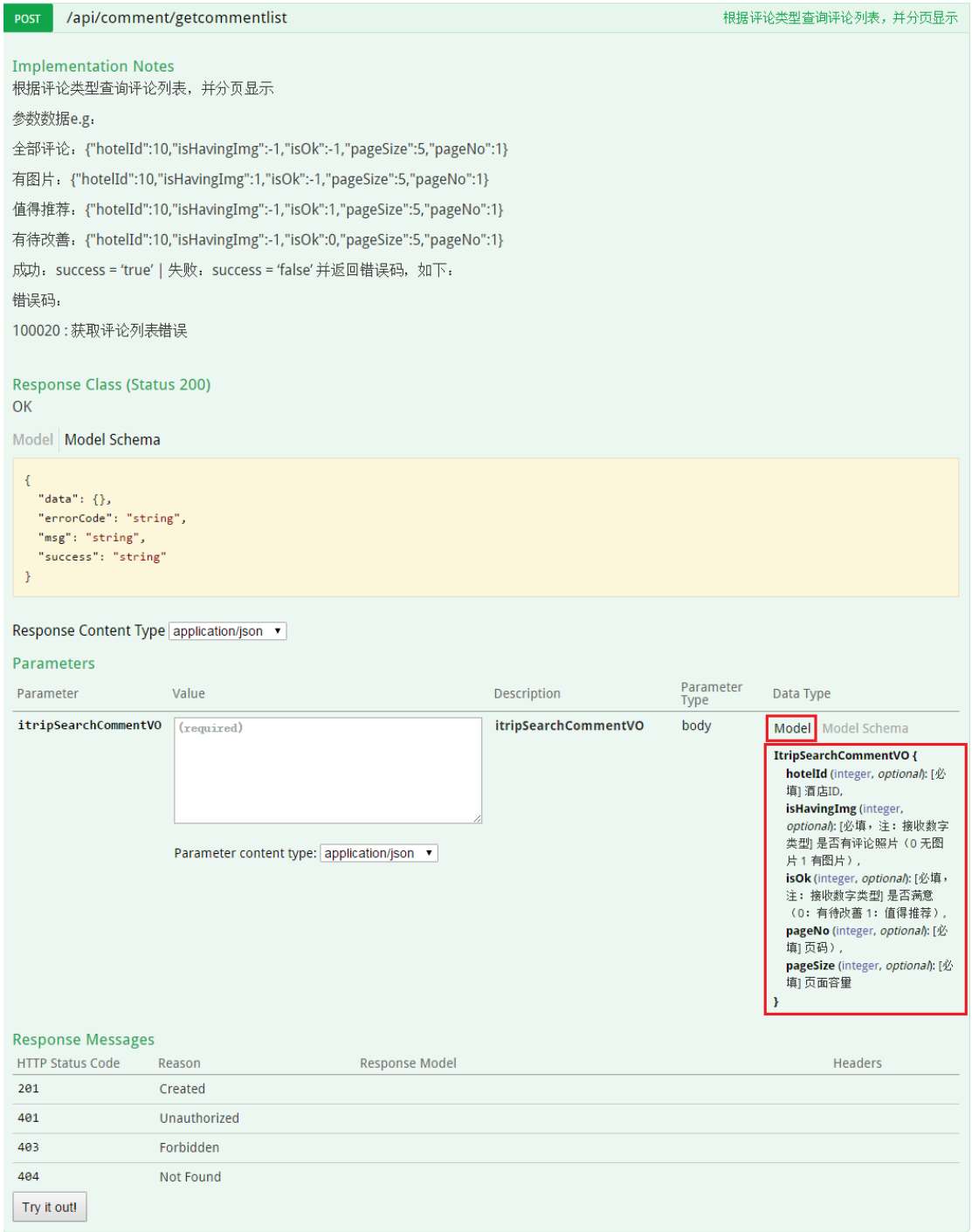

图 9

点击 Data Type 的 Model 选项，即可显示在新增的 Model 描述信息，更加清晰的展现出接口所需的各类参数标准及输入要求。