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](http://lib.csdn.net/base/17)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。

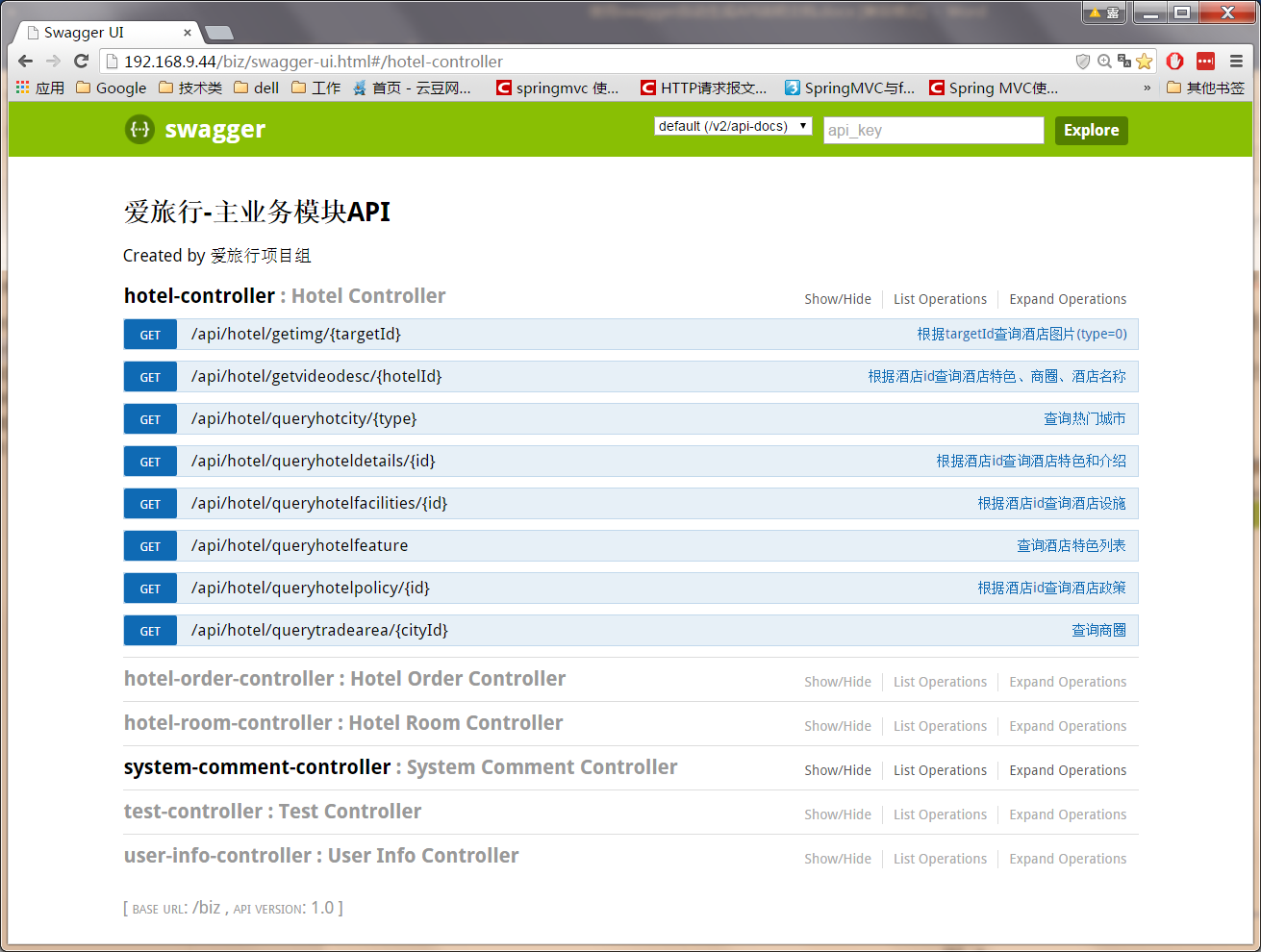


图 1

## 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> |

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

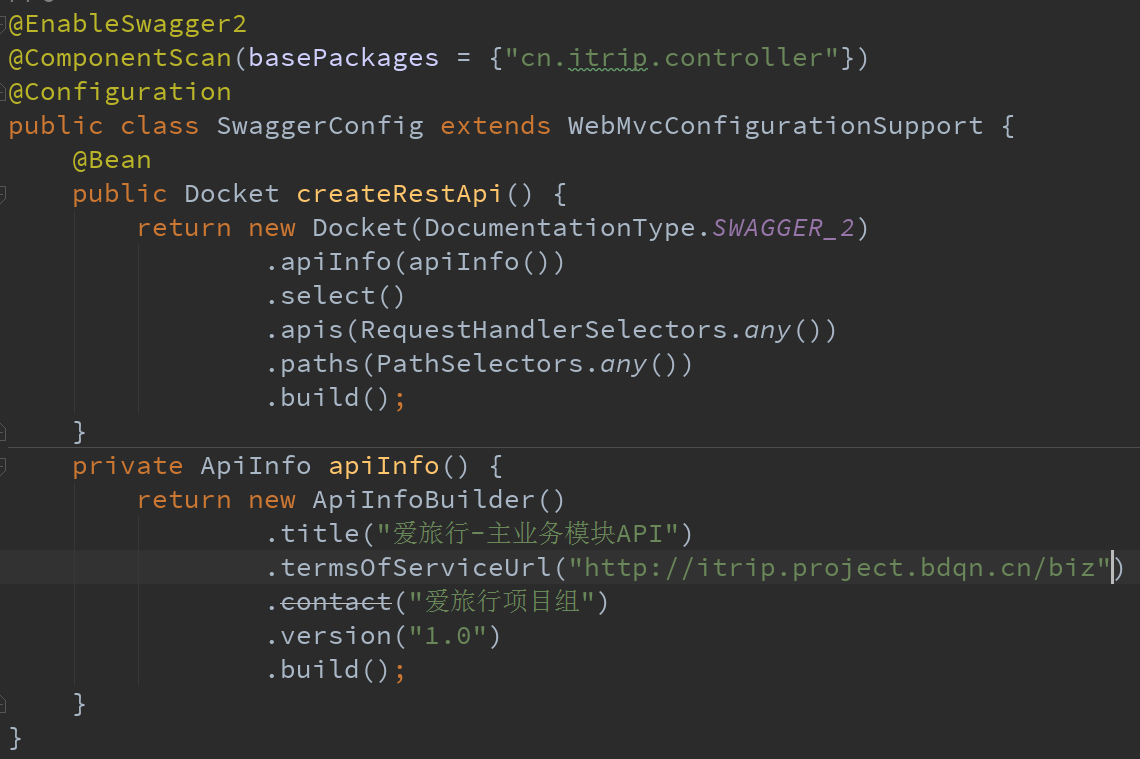


图 2

* 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）

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

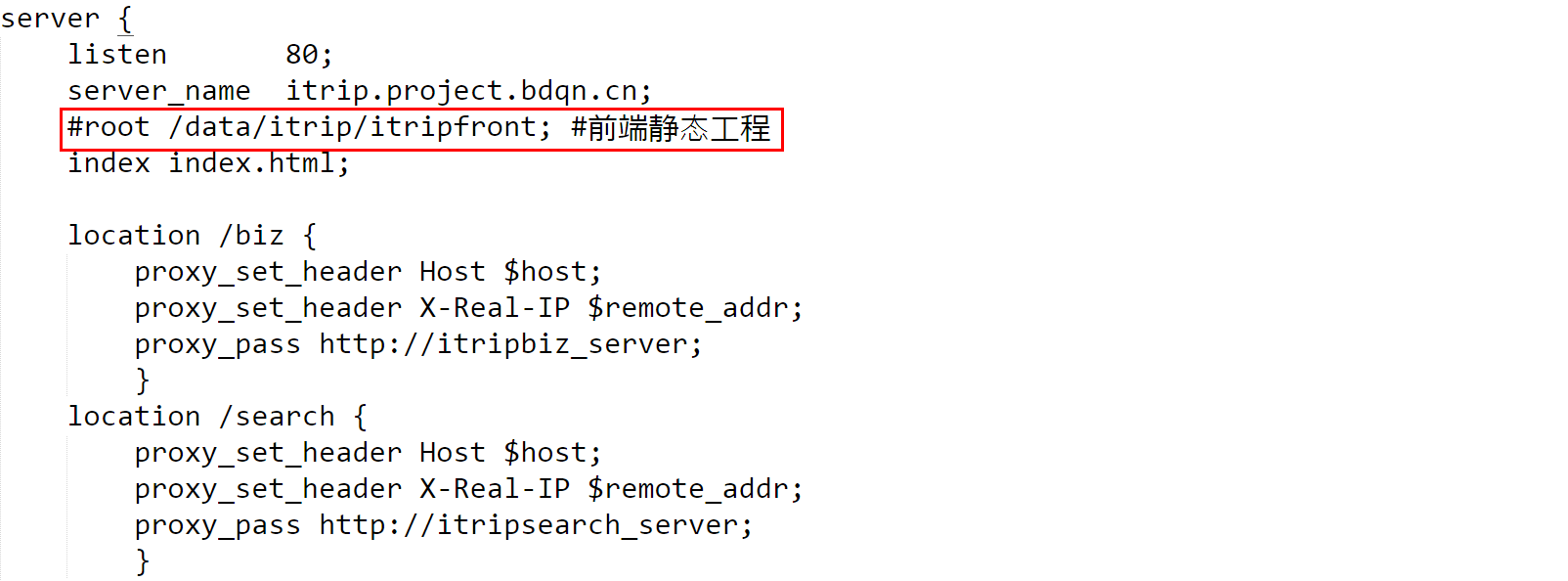
|  |
| --- |
| <!-- 使用 Swagger Restful API文档时，添加此注解 -->  <mvc:default-servlet-handler /> |

* 1. <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：



中间部分配置省略…

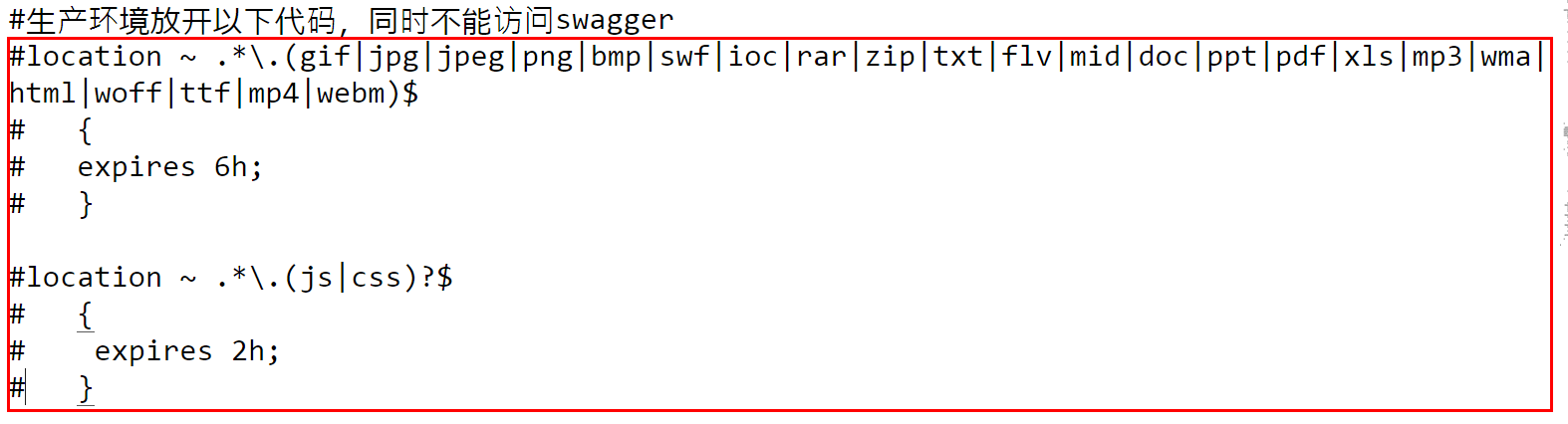


图3

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：

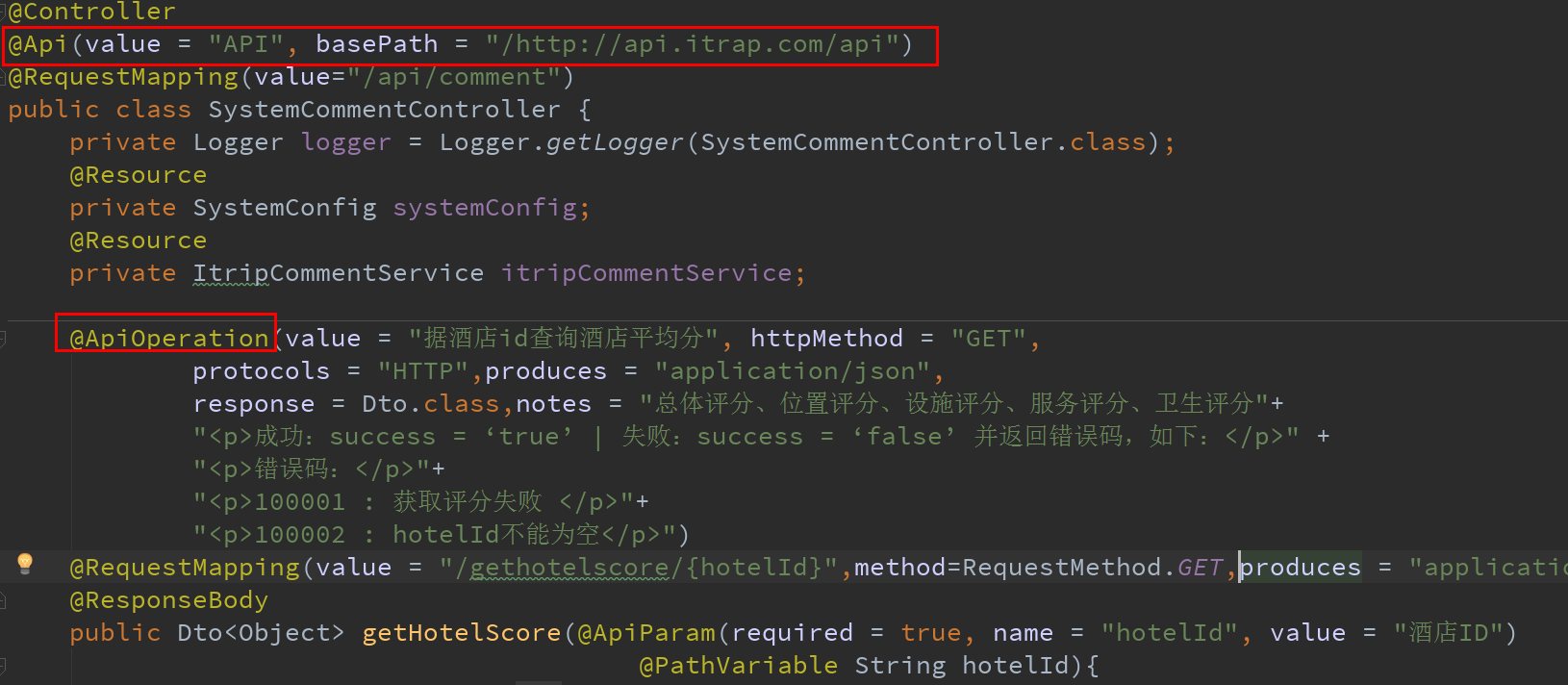


图4

* @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：

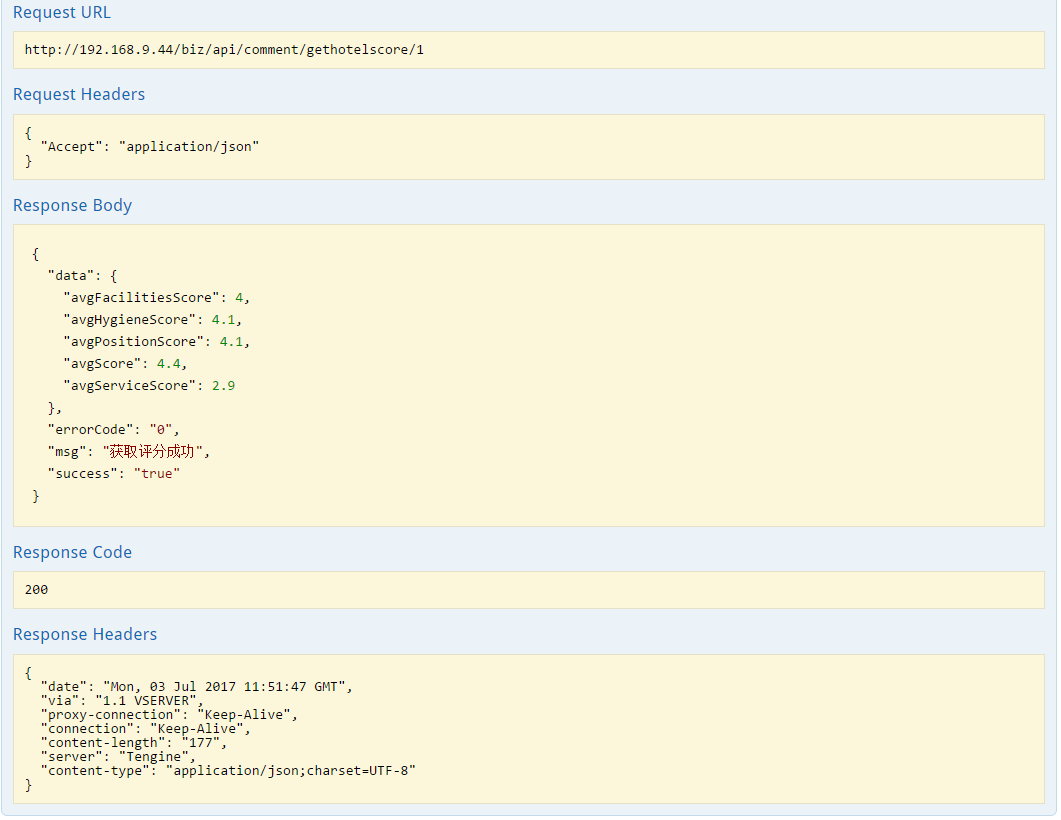


图6

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：

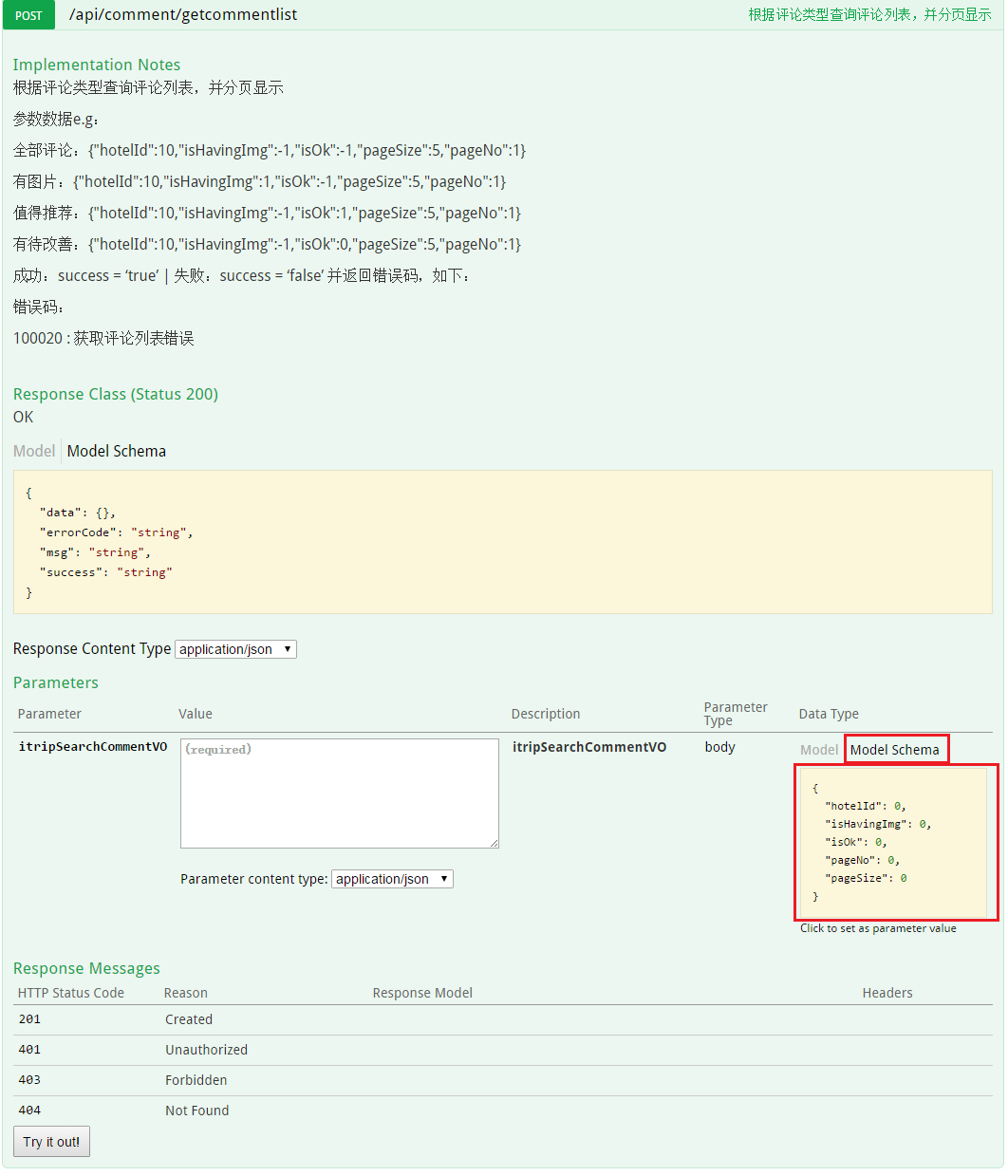


图7

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：



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：

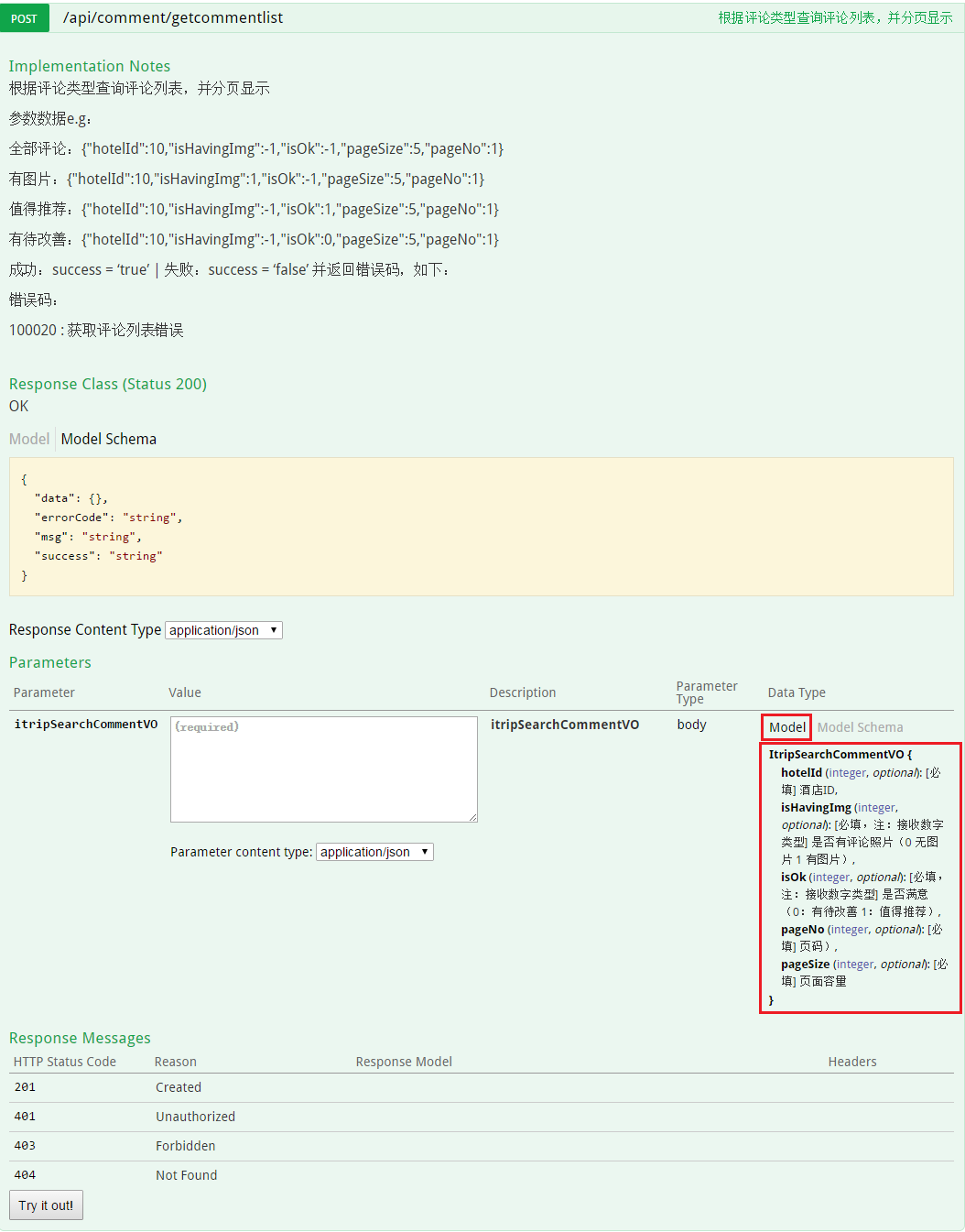


图9

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。