前后端分离后，维护接口文档基本上是必不可少的工作。一个理想的状态是设计好后，接口文档发给前端和后端，大伙按照既定的规则各自开发，开发好了对接上了就可以上线了。当然这是一种非常理想的状态，实际开发中却很少遇到这样的情况，接口总是在不断的变化之中，有变化就要去维护，做过的小伙伴都知道这件事有多么头大！还好，有一些工具可以减轻我们的工作量，Swagger2就是其中之一，至于其他类似功能但是却收费的软件，这里就不做过多介绍了。本文主要和大伙来聊下在Spring Boot中如何整合Swagger2。

**工程创建**

当然，首先是创建一个Spring Boot项目，加入web依赖，创建成功后，加入两个Swagger2相关的依赖，完整的依赖如下：

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

**Swagger2配置**

Swagger2的配置也是比较容易的，在项目创建成功之后，只需要开发者自己提供一个Docket的Bean即可，如下：

@Configuration

@EnableSwagger2

public class SwaggerConfig {

@Bean

public Docket createRestApi() {

return new Docket(DocumentationType.SWAGGER\_2)

.pathMapping("/")

.select()

.apis(RequestHandlerSelectors.basePackage("com.nvn.controller"))

.paths(PathSelectors.any())

.build().apiInfo(new ApiInfoBuilder()

.title("SpringBoot整合Swagger")

.description("SpringBoot整合Swagger，详细信息......")

.version("9.0")

.contact(new Contact("啊啊啊啊","blog.csdn.net","aaa@gmail.com"))

.license("The Apache License")

.licenseUrl("http://www.baidu.com")

.build());

}

}

这里提供一个配置类，首先通过@EnableSwagger2注解启用Swagger2，然后配置一个Docket Bean，这个Bean中，配置映射路径和要扫描的接口的位置，在apiInfo中，主要配置一下Swagger2文档网站的信息，例如网站的title，网站的描述，联系人的信息，使用的协议等等。

如此，Swagger2就算配置成功了，非常方便。

此时启动项目，输入http://localhost:8080/swagger-ui.html，能够看到如下页面，说明已经配置成功了：



**创建接口**

接下来就是创建接口了，Swagger2相关的注解其实并不多，而且很容易懂，下面我来分别向小伙伴们举例说明：

@RestController

@Api(tags = "用户管理相关接口")

@RequestMapping("/user")

public class UserController {

@PostMapping("/")

@ApiOperation("添加用户的接口")

@ApiImplicitParams({

@ApiImplicitParam(name = "username", value = "用户名", defaultValue = "李四"),

@ApiImplicitParam(name = "address", value = "用户地址", defaultValue = "深圳", required = true)

}

)

public RespBean addUser(String username, @RequestParam(required = true) String address) {

return new RespBean();

}

@GetMapping("/")

@ApiOperation("根据id查询用户的接口")

@ApiImplicitParam(name = "id", value = "用户id", defaultValue = "99", required = true)

public User getUserById(@PathVariable Integer id) {

User user = new User();

user.setId(id);

return user;

}

@PutMapping("/{id}")

@ApiOperation("根据id更新用户的接口")

public User updateUserById(@RequestBody User user) {

return user;

}

}

这里边涉及到多个API，我来向小伙伴们分别说明：

1. @Api注解可以用来标记当前Controller的功能。
2. @ApiOperation注解用来标记一个方法的作用。
3. @ApiImplicitParam注解用来描述一个参数，可以配置参数的中文含义，也可以给参数设置默认值，这样在接口测试的时候可以避免手动输入。
4. 如果有多个参数，则需要使用多个@ApiImplicitParam注解来描述，多个@ApiImplicitParam注解需要放在一个@ApiImplicitParams注解中。
5. 需要注意的是，@ApiImplicitParam注解中虽然可以指定参数是必填的，但是却不能代替@RequestParam(required = true)，前者的必填只是在Swagger2框架内必填，抛弃了Swagger2，这个限制就没用了，所以假如开发者需要指定一个参数必填，@RequestParam(required = true)注解还是不能省略。
6. 如果参数是一个对象（例如上文的更新接口），对于参数的描述也可以放在实体类中。例如下面一段代码：

@ApiModel

public class User {

@ApiModelProperty(value = "用户id")

private Integer id;

@ApiModelProperty(value = "用户名")

private String username;

@ApiModelProperty(value = "用户地址")

private String address;

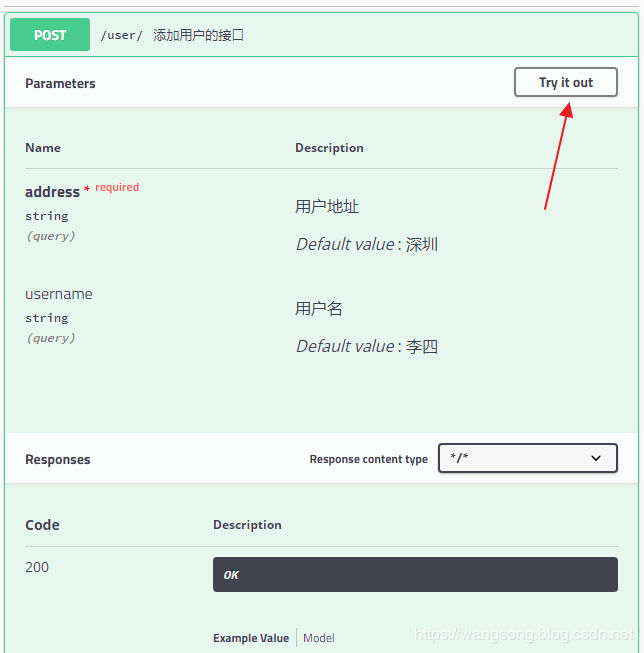
//getter/setter

}

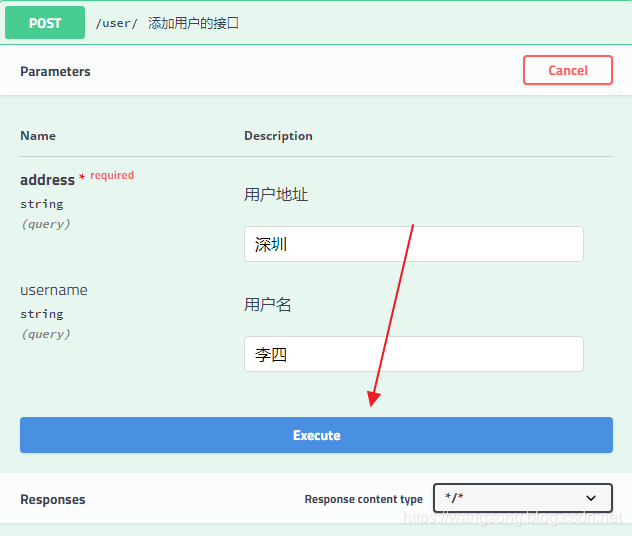
好了，经过如上配置之后，接下来，刷新刚刚打开的页面，可以看到如下效果：



可以看到，所有的接口这里都列出来了，包括接口请求方式，接口地址以及接口的名字等，点开一个接口，可以看到如下信息：



可以看到，接口的参数，参数要求，参数默认值等等统统都展示出来了，参数类型下的query表示参数以key/value的形式传递，点击右上角的Try it out，就可以进行接口测试：

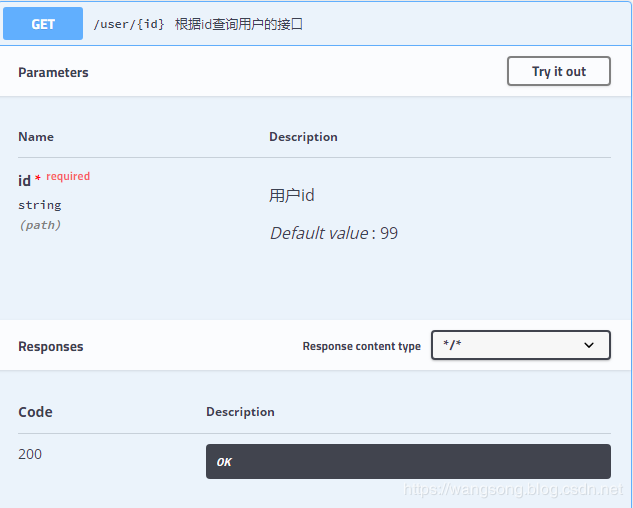


点击Execute按钮，表示发送请求进行测试。测试结果会展示在下面的Response中。

小伙伴们注意，参数类型下面的query表示参数以key/value的形式传递，这里的值也可能是body，body表示参数以请求体的方式传递，例如上文的更新接口，如下：



当然还有一种可能就是这里的参数为path，表示参数放在路径中传递，例如根据id查询用户的接口：



当然，除了这些之外，还有一些响应值的注解，都比较简单，小伙伴可以自己摸索下。

**在Security中的配置**

如果我们的Spring Boot项目中集成了Spring Security，那么如果不做额外配置，Swagger2文档可能会被拦截，此时只需要在Spring Security的配置类中重写configure方法，添加如下过滤即可：

@Override

public void configure(WebSecurity web) throws Exception {

web.ignoring()

.antMatchers("/swagger-ui.html")

.antMatchers("/v2/\*\*")

.antMatchers("/swagger-resources/\*\*");

}