第 2-9 课：使用 Swagger 2 构建 RESTful APIs

什么是 Swagger

Swagger 是一系列 RESTful API 的工具，通过 Swagger 可以获得项目的一种交互式文档，客户端 SDK 的自动生成等功能。

Swagger 的目标是为 REST APIs 定义一个标准的、与语言无关的接口，使人和计算机在看不到源码或者看不到文档或者不能通过网络流量检测的情况下，能发现和理解各种服务的功能。当服务通过 Swagger 定义，消费者就能与远程的服务互动通过少量的实现逻辑。类似于低级编程接口，Swagger 去掉了调用服务时的很多猜测。

Swagger（丝袜哥）是世界上最流行的 API 表达工具。

Swagger 是一个简单但功能强大的 API 表达工具。它具有地球上最大的 API 工具生态系统，数以千计的开发人员，使用几乎所有的现代编程语言，都在支持和使用 Swagger。使用 Swagger 生成 API，我们可以得到交互式文档，自动生成代码的 SDK 以及 API 的发现特性等。

使用 Spring Boot 集成 Swagger 的理念是，使用注解来标记出需要在 API 文档中展示的信息，Swagger 会根据项目中标记的注解来生成对应的 API 文档。Swagger 被号称世界上最流行的 API 工具，它提供了 API 管理的全套解决方案，API 文档管理需要考虑的因素基本都包含，这里将讲解最常用的定制内容。

快速上手

Spring Boot 集成 Swagger 2.X 很简单，需要引入依赖并做基础配置即可，下面我们来感受一下。

添加依赖包

```
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.8.0</version>
</dependency>
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.8.0</version>
</dependency>
```

创建 SwaggerConfig 配置类

```
@Configuration
@EnableSwagger2
public class SwaggerConfig {
}
```

在 SwaggerConfig 的类上添加两个注解：

- @Configuration，启动时加载此类
- @EnableSwagger2，表示此项目启用 Swagger API 文档

在 SwaggerConfig 中添加两个方法：

```
@Bean
public Docket api() {
    return new Docket(DocumentationType.SWAGGER_2)
            .apiInfo(apiInfo())
            .select()
            // 自行修改为自己的包路径
            .apis(RequestHandlerSelectors.basePackage("com.neo.xxx"))
            .paths(PathSelectors.any())
            .build();
}
```

此方法使用 @Bean，在启动时初始化，返回实例 Docket（Swagger API 摘要），这里需要注意的是 .apis(RequestHandlerSelectors.basePackage("com.neo.xxx")) 指定需要扫描的包路径，只有此路径下的 Controller 类才会自动生成 Swagger API 文档。

```
private ApiInfo apiInfo() {
    return new ApiInfoBuilder()
            .title("客户管理")
            .description("客户管理中心 API 1.0 操作文档")
            //服务条款网址
            .termsOfServiceUrl("http://www.ityouknow.com/")
            .version("1.0")
            .contact(new Contact("纯洁的微笑", "http://www.ityouknow.com/", "ityoukn
ow@126.com"))
            .build();
}
```

这块配置相对重要一些，主要配置页面展示的基本信息包括，标题、描述、版本、服务条款、联系方式等，查看 ApiInfo 类的源码还会发现支持 license 配置等。

```
public class ApiInfo {
    public static final Contact DEFAULT_CONTACT = new Contact("", "", "");
    public static final ApiInfo DEFAULT;
    private final String version;
    private final String title;
    private final String description;
    private final String termsOfServiceUrl;
    private final String license;
    private final String licenseUrl;
    private final Contact contact;
    private final List<VendorExtension> vendorExtensions;
    //...

}
```

以上信息皆可在此方法进行配置，也可以使用默认值。配置完成之后启动项目，在浏览器中输入网址 http://localhost:8080/swagger-ui.html，即可看到上面的配置信息，效果如下：

访问地址后，发现页面存在这样一句话：No operations defined in spec!，意思是没有找到相关的 API 内容，这是因为还没有添加对应的 Controller 信息，接下来结合代码一一介绍各个注解的使用。

Swagger 常用注解

Swagger 通过注解表明该接口会生成文档，包括接口名、请求方法、参数、返回信息等，常用注解内容如

下：

作用范围	API	使用位置
协议集描述	@Api	用于 Controller 类上
协议描述	@ApiOperation	用在 Controller 的方法上
非对象参数集	@ApiImplicitParams	用在 Controller 的方法上
非对象参数描述	@ApiImplicitParam	用在 @ApiImplicitParams 的方法里边
响应集	@ApiResponses	用在 Controller 的方法上
响应信息参数	@ApiResponse	用在 @ApiResponses 里边
描述返回对象的意义	@ApiModel	用在返回对象类上
对象属性	@ApiModelProperty	用在出入参数对象的字段上

在第 2-8 课讲解的示例项目基础上，添加 RESTful API 文档示例。

@Api 的使用

Api 作用在 Controller 类上，做为 Swagger 文档资源，该注解将一个 Controller（Class）标注为一个 Swagger 资源（API）。在默认情况下，Swagger-Core 只会扫描解析具有 @Api 注解的类，而会自动忽略其他类别资源（JAX-RS endpoints、Servlets 等）的注解。

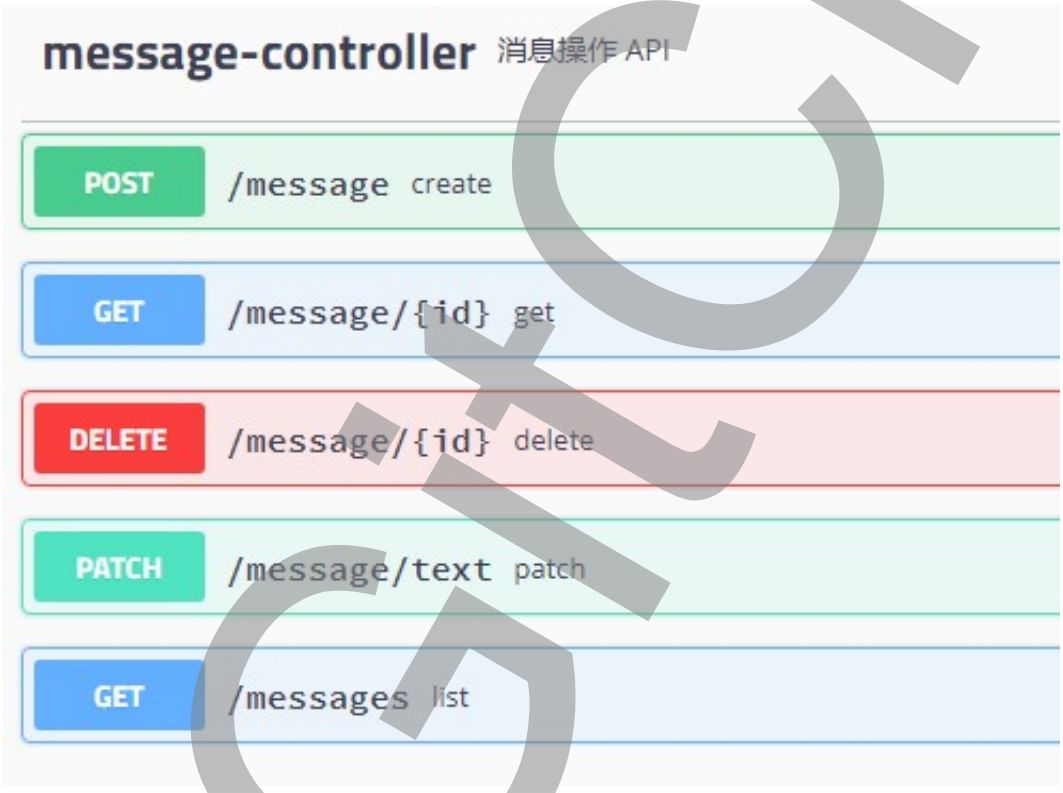
使用示例：

```
@Api(value = "消息", description = "消息操作 API", position = 100, protocols = "http
")
@RestController
@RequestMapping("/")
public class MessageController {
}
```

与 Controller 注解并列使用，属性配置如表所示：

属性名称	备注
value	url 的路径值
tags	如果设置这个值，value 的值会被覆盖
description	对 API 资源的描述
produces	For example, "application/json, application/xml"
consumes	For example, "application/json, application/xml"
protocols	Possible values: http, https, ws, wss
authorizations	高级特性认证时配置
hidden	配置为 true 将在文档中隐藏

重启项目之后，在浏览器中输入网址 http://localhost:8080/swagger-ui.html#/message-controller，可以看到如下效果：



自动将 MessageController 内的方法都添加了映射，并标明了每种方法的请求方式。

@ApiOperation 的使用

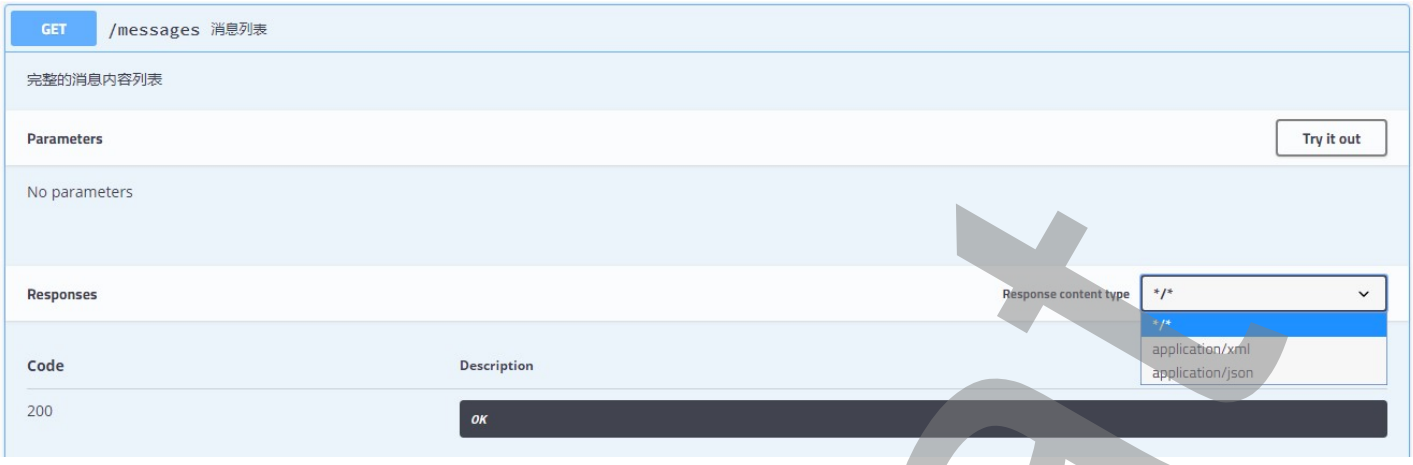
ApiOperation 定义在方法上，描述方法名、方法解释、返回信息、标记等信息。

使用示例：

```
@ApiOperation(
            value = "消息列表",
            notes = "完整的消息内容列表",
            produces="application/json, application/xml",
            consumes="application/json, application/xml",
            response = List.class)
@GetMapping(value = "messages")
public List<Message> list() {
}
```

属性名称	备注
value	url 的路径值
tags	如果设置这个值、value的 值会被覆盖
produces	For example, "application/json, application/xml"
consumes	For example, "application/json, application/xml"
protocols	Possible values: http, https, ws, wss
authorizations	高级特性认证时配置
hidden	配置为 true 将在文档中隐藏
response	返回的对象
responseContainer	这些对象是有效的 "List", "Set" or "Map"，其他无效
httpMethod	"GET"、"HEAD"、"POST"、"PUT"、"DELETE"、"OPTIONS" and "PATCH"
code	http 的状态码 默认 200
extensions	扩展属性

重启项目之后，在浏览器中输入网址 http://localhost:8080/swagger-ui.html#/message-controller/listUsingGET，可以看到如下效果：



@ApiImplicitParams 和 @ApiImplicitParam 的使用

@ApiImplicitParams 用于描述方法的返回信息，和 @ApiImplicitParam 注解配合使用；@ApiImplicitParam 用来描述具体某一个参数的信息，包括参数的名称、类型、限制等信息。

使用示例：

```
@ApiOperation(value = "添加消息", notes = "根据参数创建消息")
    @ApiImplicitParams({
            @ApiImplicitParam(name = "id", value = "消息 ID", required = true, data
Type = "Long", paramType = "query"),
            @ApiImplicitParam(name = "text", value = "正文", required = true, dataT
ype = "String", paramType = "query"),
            @ApiImplicitParam(name = "summary", value = "摘要", required = false, d
ataType = "String", paramType = "query"),
})
@PostMapping(value = "message")
public Message create(Message message) {
}
```

属性名称	备注
name	接收参数名
value	接收参数的意义描述
required	参数是否必填值为 true 或者 false
dataType	参数的数据类型只作为标志说明，并没有实际验证
paramType	查询参数类型，其值： path 以地址的形式提交数据 query 直接跟参数完成自动映射赋 body 以流的形式提交，仅支持 POST header 参数在 request headers 里边提交 form 以 form 表单的形式提交 仅支持 POST
defaultValue	默认值

重启项目之后，在浏览器中输入网址 http://localhost:8080/swagger-ui.html#/message-controller/createUsingPOST，可以看到如下效果：

@ApiResponses 和 @ApiResponse 的使用

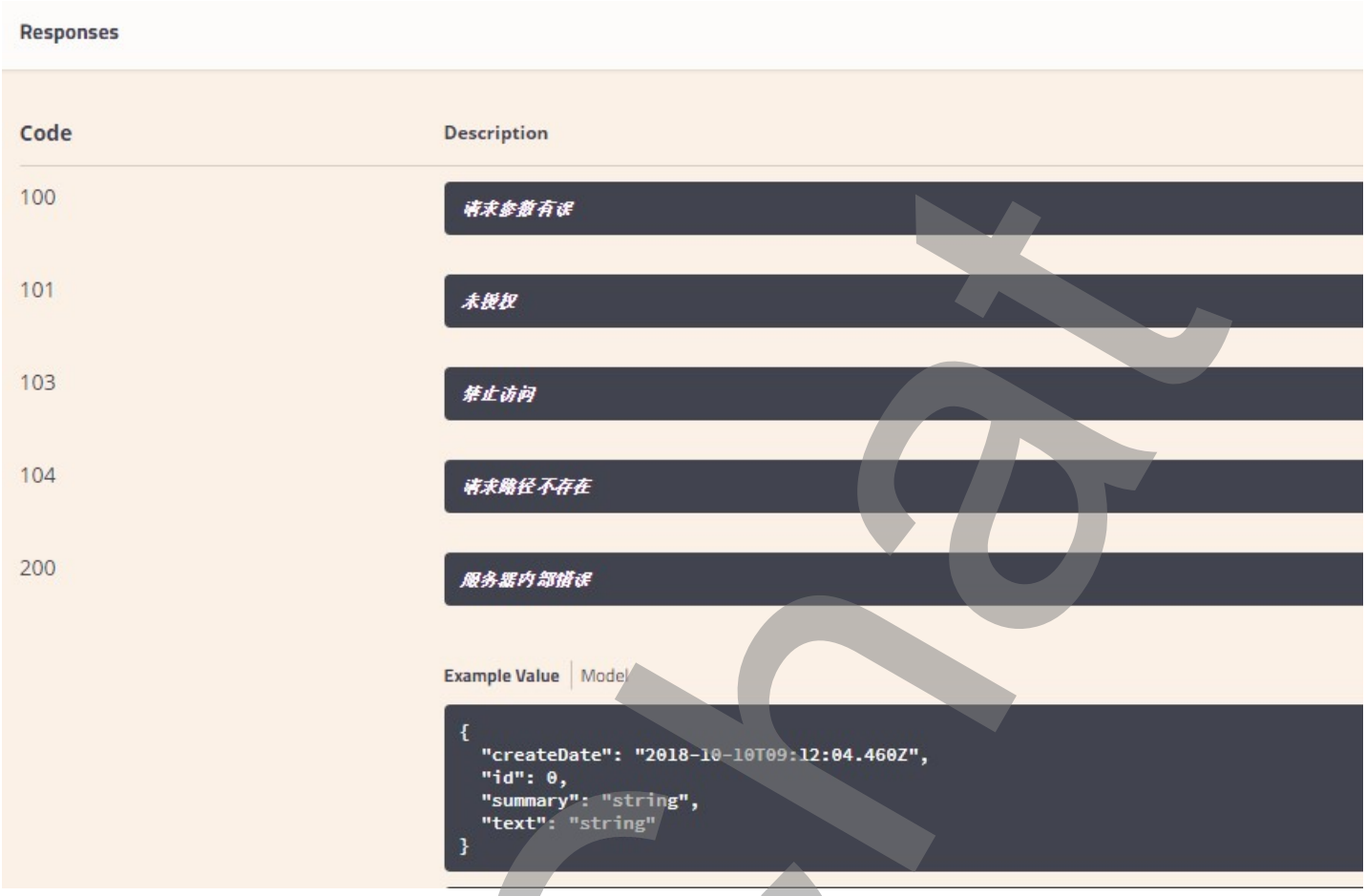
@ApiResponses 主要封装方法的返回信息和 @ApiResponse 配置起来使用，@ApiResponse 定义返回的具体信息包括返回码、返回信息等。

使用示例：

```
@ApiOperation(value = "修改消息", notes = "根据参数修改消息")
@PutMapping(value = "message")
@ApiResponses({
        @ApiResponse(code = 100, message = "请求参数有误"),
        @ApiResponse(code = 101, message = "未授权"),
        @ApiResponse(code = 103, message = "禁止访问"),
        @ApiResponse(code = 104, message = "请求路径不存在"),
        @ApiResponse(code = 200, message = "服务器内部错误")
})
public Message modify(Message message) {
}
```

属性名称	备注
code	http 的状态码
message	描述
response	默认响应类 Void
reference	参考
responseHeaders	封装返回信息
responseContainer	字符串

重启项目之后，在浏览器中输入网址 http://localhost:8080/swagger-ui.html#/message-controller/modifyUsingPUT，可以看到如下效果：



@ApiModel 和 @ApiModelProperty 的使用

在实际的项目中我们常常会封装一个对象作为返回值，@ApiModel 就是负责描述对象的信息，@ApiModelProperty 负责描述对象中属性的相关内容。

使用示例：

```
@ApiModel(description = "响应对象")
public class BaseResult<T> {
    private static final int SUCCESS_CODE = 0;
    private static final String SUCCESS_MESSAGE = "成功";

    @ApiModelProperty(value = "响应码", name = "code", required = true, example = "
" + SUCCESS_CODE)
    private int code;
    @ApiModelProperty(value = "响应消息", name = "msg", required = true, example = S
UCCESS_MESSAGE)
    private String msg;
    @ApiModelProperty(value = "响应数据", name = "data")
    private T data;
}
```

属性配置如下表所示：

属性名称	备注
value	属性描述
name	如果配置覆盖属性名称
allowableValues	允许的值
access	可以不配置
notes	没有使用
dataType	数据类型
required	是否为必传参数
position	显示的顺序位置
hidden	是否因此
example	举例
readOnly	只读
reference	引用

这样我们在 Controller 中封装返还信息时就可以这样操作：

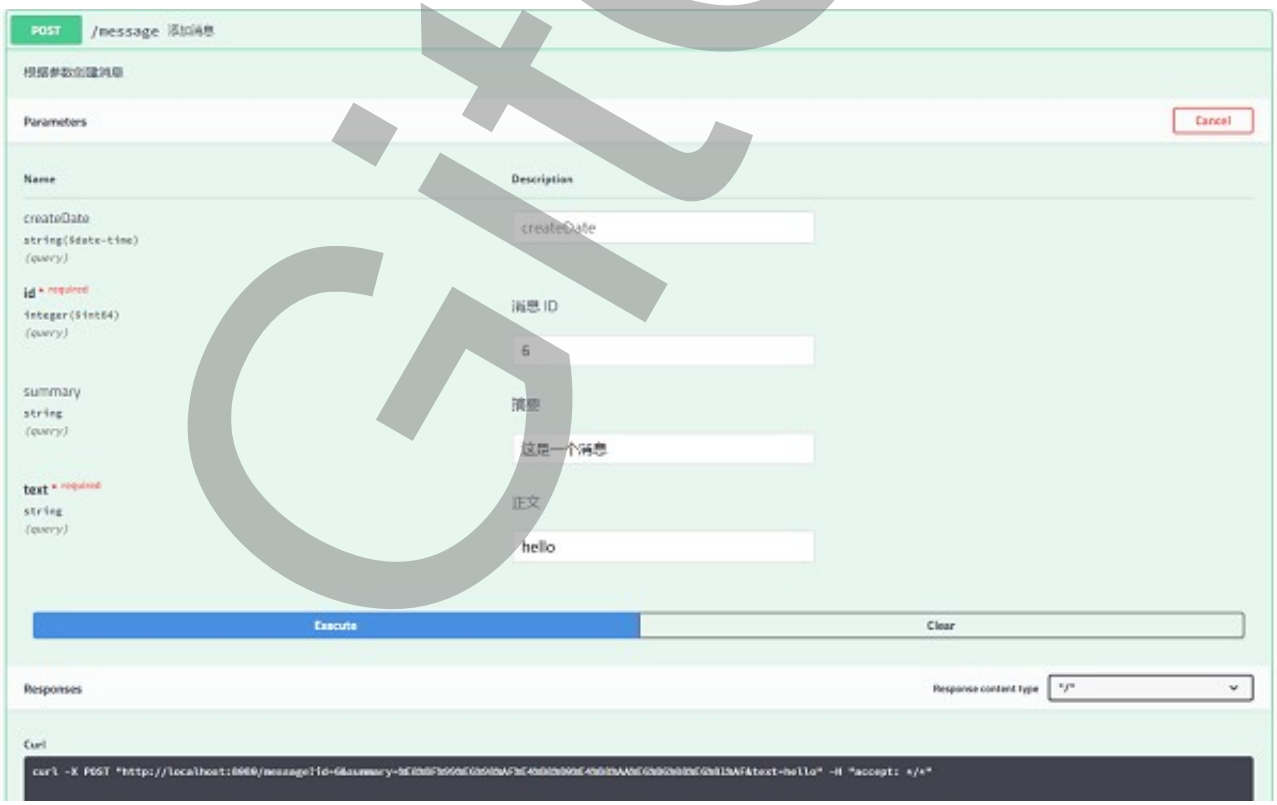
```
@PatchMapping(value="/message/text")
public BaseResult<Message> patch(Message message) {
    Message messageResult=this.messageRepository.updateText(message);
    return BaseResult.successWithData(messageResult);
}
```

重启项目之后，在浏览器中输入网址 http://localhost:8080/swagger-ui.html，点解页面 Models 折叠项，可以看到如下效果：

以上就是在项目中最常用的一些注解，灵活地利用这些注解就可以自动构建出清晰的 API 文档。

Try it out

使用 Swagger 创建的在线 API 还有一个非常强大的功能，可以在页面直接测试接口的可用性，这样在前端和后端接口调试出现问题时，可以非常方便地利用此功能进行接口验证。在上面参数讲解过程中，我们发现每个接口描述右侧都有一个按钮 try it out，单击 try it out 按钮即可进入表单页面，如下：



在表单页面添加相关字段后，单击“Execute”按钮就会将请求发送到后台，从而进行接口验证，通过按钮下面的命令可以看出，实际上是使用了 curl 命令进行的 post 测试：

```
curl -X POST "http://localhost:8080/message?id=6&summary=%E8%BF%99%E6%98%AF%E4%B8%80%E4%B8%AA%E6%B6%88%E6%81%AF&text=hello" -H "accept: */*"
```

在后端调整 Swagger 方法上对应参数，即可看到 curl 命令参数的变化。

总结

通过这节课的学习我们掌握了在 Spring Boot 项目中使用 Swagger，利用 Swagger 的相关注解可以容易地构建出丰富的 API 文档。使用 Swagger 之后可以帮助生成标准的 API 说明文档，避免接口交互中的低效沟通问题，Swagger 做为强大的 API 生成框架其实还有更多的功能，大家有机会可以在线下继续学习。

> 点击这里下载源码。