# Swagger



## 介绍

Swagger是一个规范和完整的框架，用于生成、描述、调用和可视化 RESTful 风格的 Web 服务。总体目标是使客户端和文件系统作为服务器以同样的速度来更新。文件的方法，参数和模型紧密集成到服务器端的代码，允许API来始终保持同步。

在项目开发中，根据业务代码自动生成API文档，给前端提供在线测试，自动显示JSON格式，方便了后端与前端的沟通与调试成本。

Swagger有一个缺点就是侵入性模式，必须配置在具体的代码里。

## Swagger使用

- 在pom.xml文件中添加swagger相关依赖

```
        <!-- swagger 依赖 -->
        <!-- API获取的包 -->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
        </dependency>
        <!-- 官方给出的一个ui界面，这个界面可以自定义，默认是官方的 -->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.9.2</version>
        </dependency>
        <!-- 测试数据以JSON格式返回的依赖包 -->
        <dependency>
            <groupId>com.github.xiaoymin</groupId>
            <artifactId>swagger-bootstrap-ui</artifactId>
            <version>1.6</version>
        </dependency>
```

- 配置swagger

新建Swagger配置类，需要特别注意的是Swagger scan base package,这是扫描注解的配置，即你的API接口位置，对前端提供服务接口的位置。

```
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.builders.ParameterBuilder;
import springfox.documentation.builders.PathSelectors;
import springfox.documentation.builders.RequestHandlerSelectors;
import springfox.documentation.schema.ModelRef;
import springfox.documentation.service.Contact;
import springfox.documentation.service.Parameter;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import java.util.ArrayList;
import java.util.List;
```

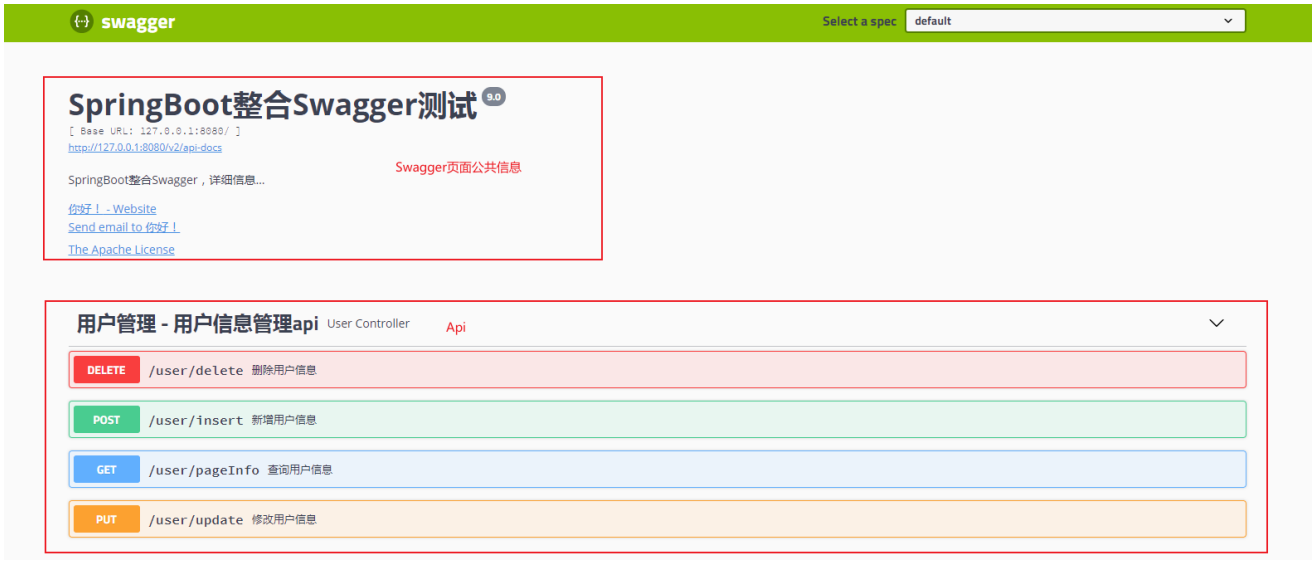
```
@Configuration
@EnableSwagger2
public class Swagger2Config {

    @Bean
    public Docket createRestApi() {
        ParameterBuilder tokenPar = new ParameterBuilder();
        // 默认参数，添加上后，所有的接口都会有一个公共参数，不需要在每个接口单独配置
        List<Parameter> pars = new ArrayList<Parameter>();
        tokenPar.name("token").description("令牌")
                .modelRef(new ModelRef("string")).parameterType("query").required(false).build();
        pars.add(tokenPar.build());
        return new Docket(DocumentationType.SWAGGER_2)
                .pathMapping("/")
                .select()
                // 描注解的配置
                .apis(RequestHandlerSelectors.basePackage("com.demo.controller"))
                .paths(PathSelectors.any())
                .build().apiInfo(new ApiInfoBuilder()
                        .title("SpringBoot整合Swagger测试")
                        .description("SpringBoot整合Swagger，详细信息......")
                        .version("9.0")
                        .contact(new Contact("你好！","blog.csdn.net","aaa@gmail.com"))
                        .license("The Apache License")
                        .licenseUrl("http://www.baidu.com")
                        .build()).globalOperationParameters(pars);
    }

}
```

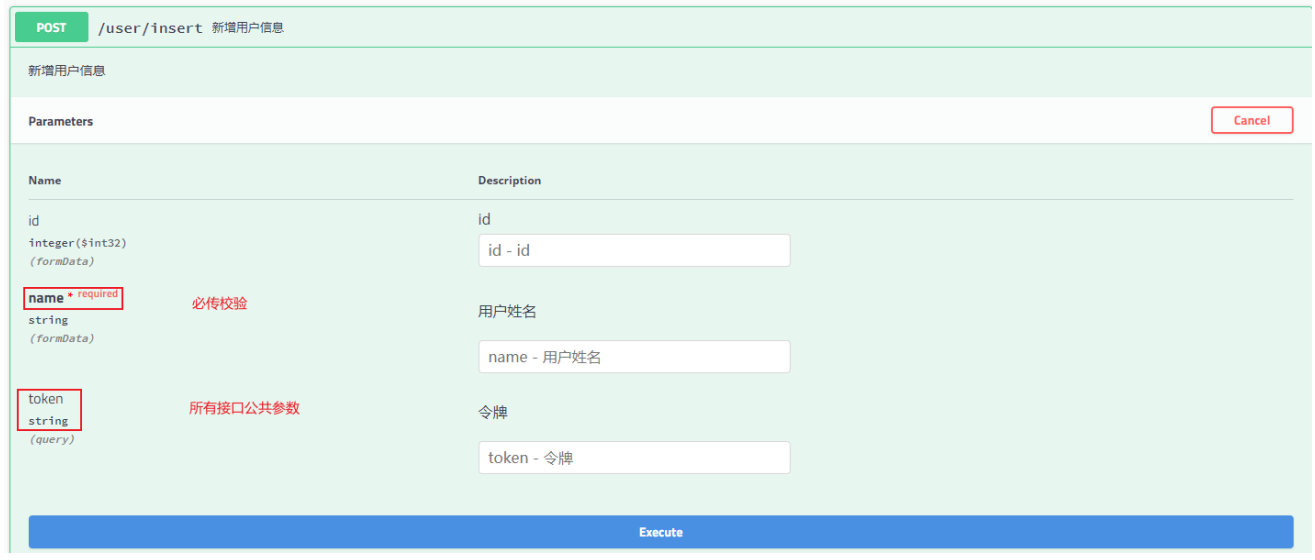
- 撰写Controller ( UserController )

```
import io.swagger.annotations.Api;
import io.swagger.annotations.ApiImplicitParam;
import io.swagger.annotations.ApiImplicitParams;
import io.swagger.annotations.ApiOperation;
import org.springframework.web.bind.annotation.*;

/* 类注解 */

@RestController
@RequestMapping("/user")
@Api(tags = "用户管理 - 用户信息管理api")
public class UserController {

    @ApiOperation(value = "新增用户信息", notes = "新增用户信息")
    @ApiImplicitParams({
            @ApiImplicitParam(name="name", value="用户姓名", dataType = "String", required=true,
paramType="form"),
            @ApiImplicitParam(name="id", value="id", dataType = "int", required=false, paramType="form")
    })
    @PostMapping("/insert")
    public String insert() {
        return "success";
    }

    @ApiOperation(value = "删除用户信息", notes = "删除用户信息")
    @DeleteMapping("/delete")
    public String delete(@RequestParam String id) {
        return "success";
    }

    @ApiOperation(value = "修改用户信息", notes = "修改用户信息")
```

```
    @PutMapping("/update")
    public String update() {
        return "success";
    }

    @ApiOperation(value = "查询用户信息", notes = "查询用户信息")
    @GetMapping("/pageInfo")
    public String pageInfo() {
        return "success";
    }
}
```

- 访问




## Swagger注解说明

- @Api：用在请求的类上，说明该类的作用

```
@Api：用在请求的类上，说明该类的作用
      tags="说明该类的作用"
      value="该参数没什么意义，所以不需要配置"
```

示例：

```
@Api(tags = "用户管理 - 用户信息管理api")
public class UserController {
    ...
}
```

- @ApiOperation：用在请求的方法上，说明方法的作用

```
@ApiOperation: "用在请求的方法上，说明方法的作用"
     value="说明方法的作用"
     notes="方法的备注说明"
```

示例：

```
@ApiOperation(value = "删除用户信息", notes = "删除用户信息")
@DeleteMapping("/delete")
public String delete(@RequestParam String id) {
    return "success";
}
```

- @ApiImplicitParams：用在请求的方法上，包含一组参数说明

```
@ApiImplicitParams：用在请求的方法上，包含一组参数说明
      @ApiImplicitParams 注解用于指定一个请求参数的配置信息
          name：参数名
          value：参数的汉字说明、解释
          required：参数是否必须传
          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）
          dataType：参数类型，默认String，其它值dataType="Integer"
          defaultValue：参数的默认值
```

示例：

```
@ApiOperation(value = "新增用户信息", notes = "新增用户信息")
@ApiImplicitParams({
       @ApiImplicitParam(name="name", value="用户姓名", dataType = "String", required=true,
paramType="form"),
       @ApiImplicitParam(name="id", value="id", dataType = "int", required=false, paramType="form")
})
@PostMapping("/insert")
public String insert() {
    return "success";
}
```

- @ApiResponses：用于请求的方法上，表示一组响应

```
@ApiResponses：用于请求的方法上，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
    人话：设置错误码信息
```

示例：

```
@ApiOperation(value = "select1请求",notes = "多个参数，多种的查询参数类型")
@ApiResponses({
    @ApiResponse(code=400,message="请求参数没填好"),
    @ApiResponse(code=404,message="请求路径没有或页面跳转路径不对")
})
```

- @ApiModel：用于响应类上，表示一个返回响应数据的信息

```
@ApiModel：用于响应类上，表示一个返回响应数据的信息
            （这种一般用在post创建的时候，使用@RequestBody这样的场景，
            请求参数无法使用@ApiImplicitParam注解进行描述的时候）
    @ApiModelProperty：用在属性上，描述响应类的属性
```

示例：

```
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;

import java.io.Serializable;

@ApiModel(description= "返回响应数据")
public class RestMessage implements Serializable{

    @ApiModelProperty(value = "是否成功")
    private boolean success=true;
    @ApiModelProperty(value = "返回对象")
    private Object data;
    @ApiModelProperty(value = "错误编号")
    private Integer errCode;
    @ApiModelProperty(value = "错误信息")
    private String message;

    /* getter/setter */
}
```

## 生成静态HTML

- 引入依赖

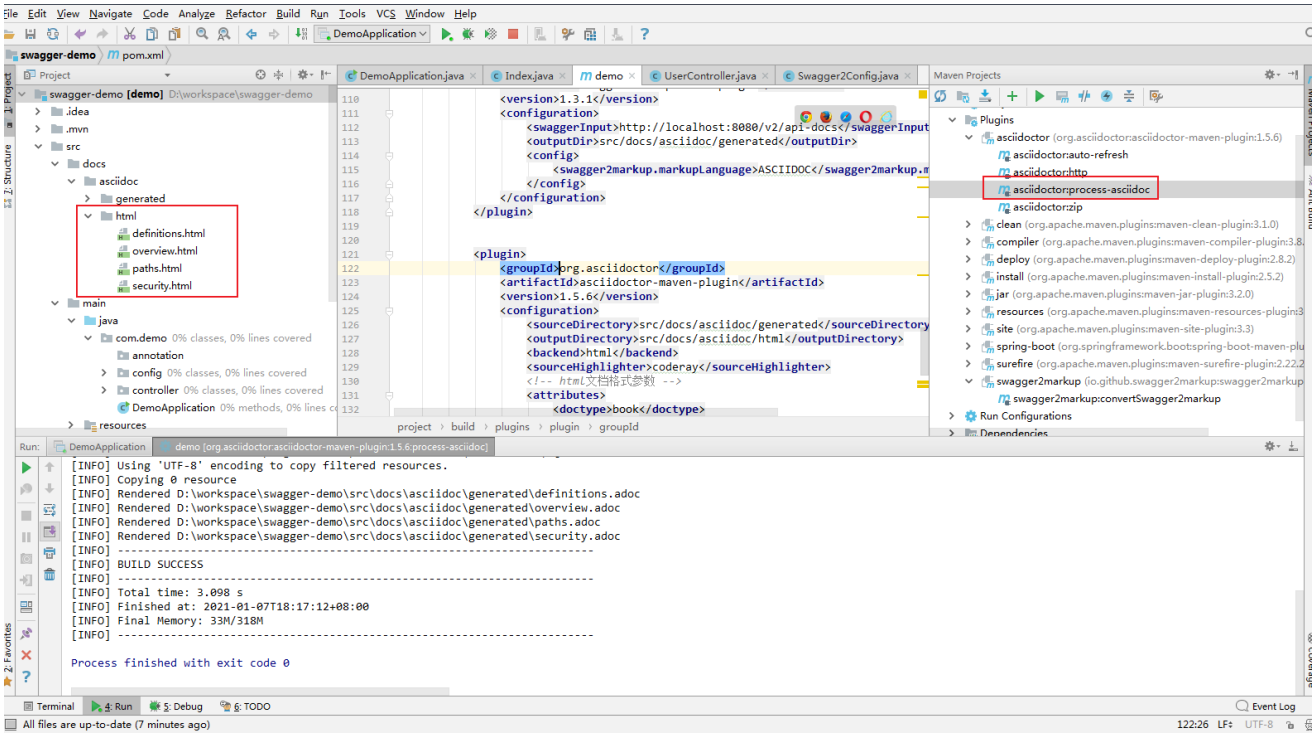
```
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.9.2</version>
            <exclusions>
                <exclusion>
                    <groupId>io.swagger</groupId>
                    <artifactId>swagger-annotations</artifactId>
                </exclusion>
                <exclusion>
                    <groupId>io.swagger</groupId>
                    <artifactId>swagger-models</artifactId>
                </exclusion>
            </exclusions>
        </dependency>

        <dependency>
            <groupId>io.swagger</groupId>
            <artifactId>swagger-annotations</artifactId>
            <version>1.5.21</version>
        </dependency>

        <dependency>
            <groupId>io.swagger</groupId>
            <artifactId>swagger-models</artifactId>
            <version>1.5.21</version>
        </dependency>

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

```
            <version>2.9.2</version>
        </dependency>

        <dependency>
            <groupId>io.github.swagger2markup</groupId>
            <artifactId>swagger2markup</artifactId>
            <version>1.3.3</version>
        </dependency>
```

- 引入插件

```
            <plugin>
                <groupId>org.asciidoctor</groupId>
                <artifactId>asciidoctor-maven-plugin</artifactId>
                <version>1.5.6</version>
                <configuration>
                    <sourceDirectory>src/docs/asciidoc/generated</sourceDirectory>
                    <outputDirectory>src/docs/asciidoc/html</outputDirectory>
                    <backend>html</backend>
                    <sourceHighlighter>coderay</sourceHighlighter>
                    <attributes>
                        <toc>left</toc>
                    </attributes>
                </configuration>
            </plugin>
```

- 执行命令该插件的asciidoctor:process-asciidoc命令




详细文档参考：

https://www.jianshu.com/p/571fe170889d/