1、Swagger 作用：在前后端分离架构下，前端工程师通过 Swagger 实时**跟踪后端程序最新 API**，缩短前后端沟通成本，提高开发效率。

2、常用注解

1）**@Api**(tags＝“xxx 模块说明”)：作用在**模块类上**；

2）**@ApiParam**(“xxx 参数说明”)：作用在**参数、方法和字段上**，required 为 true 表示必传参数；

3）**@ApiModel**(“xxxPOJO 说明”)：**作用在实体类上**；

4）**@ApiOperation**(“xxx 接口说明”)：作用在**接口方法上**；

5）**@ApiModelProperty**(value＝“属性说明”, hidden=true, required=false)：作用在**类方法和属性**上，hidden 为 true 可以隐藏该属性，required 为 true 表示添加或修改数据时必填参数。

3、导入相关依赖

```
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger2</artifactId>
    <version>2.9.2</version>
</dependency>
<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->
<dependency>
    <groupId>io.springfox</groupId>
    <artifactId>springfox-swagger-ui</artifactId>
    <version>2.9.2</version>
</dependency>
```

4、配置 swagger，写一个 config 文件

1）@EnableSwagger2 开启 Swagger2；

2）配置 swagger 的 Docket 的 bean 实例；

```
@Configuration
@EnableSwagger2          //开启Swagger2
public class SwaggerConfig {

    //配置了Swagger的Docket的bean实例
    @Bean
    public Docket docket(){
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo());
    }
```

3）配置 swagger 的基本信息 apiInfo；

```
//配置Swagger信息=apiInfo
private ApiInfo apiInfo(){
    //作者信息
    Contact contact = new Contact( name: "秦疆",  url: "https://blog.kuangstudy.com/",  email: "24736743@qq.com")
    return new ApiInfo(
            title: "狂神的SwaggerAPI文档",
            description: "即使再小的帆也能远航",
            version: "v1.0",
            termsOfServiceUrl: "https://blog.kuangstudy.com/",
           contact,
            license: "Apache 2.0",
            licenseUrl: "http://www.apache.org/licenses/LICENSE-2.0",
           new ArrayList()
    );
}
```

4）配置扫描接口 Docket.select()

```
    //配置了Swagger的Docket的bean实例
    @Bean
    public Docket docket(){
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                //RequestHandlerSelectors，配置要扫描接口的方式
                //basePackage:指定要扫描的包
                //any():扫描全部
                //none():不扫描
                //withClassAnnotation：扫描类上的注解，参数是一个注解的反射对象
                //withMethodAnnotation：扫描方法上的注解

.apis(RequestHandlerSelectors.basePackage("com.kuang.swagger.controller"))
                //paths()。过滤什么路径
                .paths(PathSelectors.ant("/kuang/**"))
                .build();
    }
```

5）配置是否启动 swagger；.enable(true)

```
    //配置了Swagger的Docket的bean实例
    @Bean
    public Docket docket(){
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .enable(false)//enable是否启动Swagger，如果为False，则Swagger不能再浏览器中访问
                .select()

.apis(RequestHandlerSelectors.basePackage("com.kuang.swagger.controller"))
                //.paths(PathSelectors.ant("/kuang/**"))
                .build();
    }
```

【补充】swagger 在生产环境中使用，在发布时不使用，怎么设置？

1>设置要显示的 swagger 环境；

2>通过 environment.acceptsProfiles 判断是否处于设定的环境中；

3>如果处于则 enable(true)开启 swagger。

```
@Configuration
@EnableSwagger2          //开启Swagger2
public class SwaggerConfig {

    //配置了Swagger的Docket的bean实例
    @Bean
    public Docket docket(Environment environment){

        //设置要显示的Swagger环境
        Profiles profiles = Profiles.of("dev","test");
        //通过environment.acceptsProfiles判断是否处在自己设定的环境当中
        boolean flag = environment.acceptsProfiles(profiles);
        System.out.println(flag);

        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .enable(flag)//enable是否启动Swagger，如果为False，则Swagger不能再浏览器中访问
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.kuang.swagger.controller"))
                //.paths(PathSelectors.ant("/kuang/**"))
                .build();
    }
```

6）配置 API 文档的分组；.groupName(“wy”)

7）如何配置多个分组；

创建多个 Docket 的 bean 实例，并设置不同的分组即可。

8）访问： http://localhost:8080/swagger-ui.html

5、knife4j 是对 swagger 的增强，swagger-ui 是其前身；

将之前的依赖去掉，导入下面这个依赖即可

```
<dependency>
    <groupId>com.github.xiaoymin</groupId>
    <artifactId>knife4j-spring-boot-starter</artifactId>
    <version>3.0.3</version>
</dependency>
```

访问：http://localhost:8080/doc.html