不用写Swagger注解的Springdoc实战及问题处理

哇塞！居然还有这东西~

感谢群里的小朋友给我安利这玩意！

没错，它就是springdoc，有了它，我就再也不用写Swagger注解了！

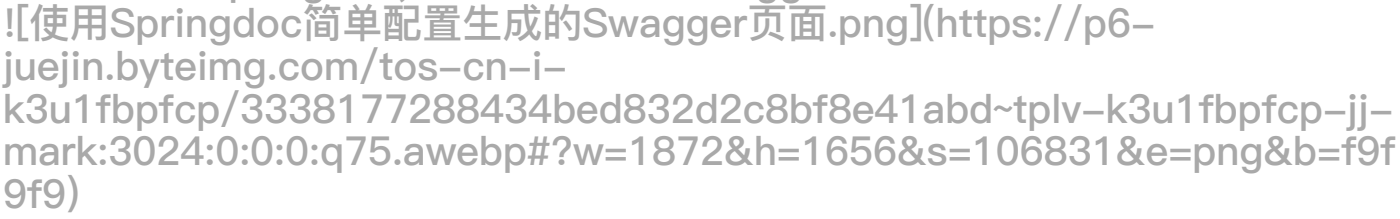
具体介绍咱就不去废话了，大伙有空网上一搜一大堆。我结合两天的配置实战，围绕以下几点，为大家作个简单介绍和具体的食用方式进行分享：

* 用Springdoc能实现什么
* 怎么使用
+ 前置要求
+ 最简单的使用配置
* 一些问题的解决
+ lombok – 引入mapstruct
+ Unable to render this definition异常 – 调整WebMvc的自定义Convert优先级

用Springdoc能实现什么

它最吸引我的地方，就是不用再写Swagger注解了，可以直接根据注释生成文档，这对于我这种手写党来说，无疑是一种解放。

而且，Swagger的页面访问依然正常访问，因为其底层就有Swagger，历史的Swagger配置也不受影响，依然可用，所以升级过渡非常简单。

以下是使用Springdoc，并且零配置的SwaggerUI的访问页面：


怎么使用

使用分为两步：

1. 前置要求
2. 最简单的使用配置

前置要求

唯一要求：**需要JDK17或以上**

配置

网上很多文章都借用大佬的ruoyi-vue-plus作为指南，或者直接就把其代码抄了出来复写一篇。不得不说，大佬的技术扎实，理解深入，但个人觉得过于繁琐，最终经过我试验得出的结论：如无特殊要求，Springdoc的使用可谓简单至极，仅需在maven的pom文件中增加以下依赖即可(目前最新版是2.5.0)：

```
<dependency>
    <groupId>org.springdoc</groupId>
    <artifactId>springdoc-openapi-starter-webmvc-ui</artifactId>
    <version>2.5.0</version>
</dependency>
```

而代码中的写法以下提供一个示例：

这是Controller

```
/**
 * Desc: TestCtr
 * ClassPath: com.example.springdocdemo.controller.TestController
```

```
 *
 * @since 2024/6/6 14:49
 */
@RequestMapping(”test”)
@RestController
public class TestController {
    /**
     * 试验
     *
     * @param query 查询
     * @return {@link Vo }
     */
    @PostMapping
    public Vo test(@RequestBody Query query) {
        var test = new Vo();
        test.setStr(”tes”);
        test.setALong(1L);
        return test;
    }
}
```

这是Query

```
/**
 * 查询
 *
 * @author deep.han
 * @date 2024/06/06
 */
@Getter
@Setter
public class Query {
    /**
     * 姓名
     */
    @NotNull(message = ”姓名不能为空”)
    private String name;
    /**
     * 年龄
     */
    @Min(value = 1, message = ”年龄不能小于1”)
    private Integer age;
}
```

```

#### 这是VO

```
/**
 * vo
 *
 * @author deep.han
 * @date 2024/06/06
 */
@Getter
@Setter
public class Vo {
    /**
     * str
     */
    private String str;
    /**
     * 一个长
     */
    private Long aLong;
}
```

#### 引入therapi

大家可以对比一下上面的Swagger访问页面图片，可以发现此时的Swagger页面对Validation的注解是可以识别的，但无论是类、函数、字段的注释说明却不能展示出来，此时需要在pom文件中增加引入依赖（官方教程也有教）：

```
/* 在dependencies中引入以下依赖 */
<dependency>
    <groupId>com.github.therapi</groupId>
    <artifactId>therapi-runtime-javadoc</artifactId>
    <version>0.15.0</version>
</dependency>

/* 在插件中引入以下插件 */
<plugin>
```
```

```
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
        <annotationProcessorPaths>
            <path>
                <groupId>com.github.therapi</groupId>
                <artifactId>therapi-runtime-javadoc-scribe</artifactId>
                <version>0.15.0</version>
            </path>
        </annotationProcessorPaths>
    </configuration>
</plugin>
```

引入依赖后，重新启动项目，加载Swagger页面，就能出现第一张图片的效果了。

如果有使用**Apifox**(推荐，识别比较全面)等，也可以直接使用链接进行导入，如当前本地示例项目的：

```
localhost:8080/v3/api-docs
```

此外，对于Swagger及api-docs的访问地址，都是可以配置的，可以参考：[OpenAPI 3 Library for spring-boot (springdoc.org)](http://cxyroad.com/ "https://springdoc.org/#javadoc-support")

具体的yml配置参数可以参考：[OpenAPI 3 Library for spring-boot (springdoc.org)](http://cxyroad.com/ "https://springdoc.org/#properties")

此处仅提供一个简单的而又比较实用的配置：

```
springdoc:
  api-docs:
    enabled: true # 是否启用, 默认是 true
```

```
    path: /api-docs # 默认是 /v3/api-docs
  swagger-ui:
    path: /swagger-ui.html # 默认是 /swagger-ui.html
    enabled: true # 是否启用, 默认是 true
  group-configs:  # 大伙可以根据需要进行分组，对Swagger访问页面及api都能生效
    - group: "Default" # 分组名称
      packages-to-scan:
        - com.example.demo.controller # 指定扫描的包路径
```

以上就是最简单的一个使用！

一些问题的解决

lombok - 引入mapstruct

当然，大家也能看到了，我是有使用lombok的，所以如果你也使用lombok，则需要引入额外的依赖，否则将出现lombok插件失效的情况：

```
// 异常处的代码
var test = new Vo();
test.setStr("tes"); //此处异常，无法加载lombok生成的set
test.setALong(1L);

D:\**\controller\TestController.java:27:13
java: 找不到符号
  符号:   方法 setStr(java.lang.String)
  位置: 类型为com.example.springdocdemo.model.Vo的变量 test
```

需要在pom文件的maven插件中增加以下配置（说实话，以下这个也是参考了ruoyi-vue-plus上的）：

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
```

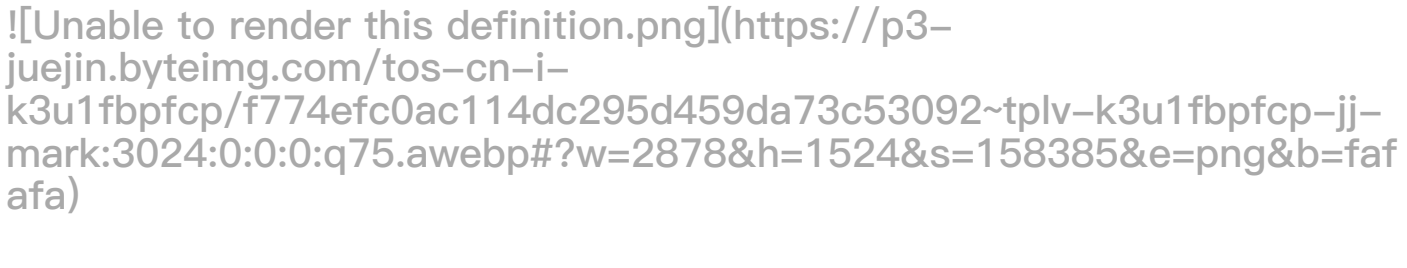
```
    <artifactId>maven-compiler-plugin</artifactId>
    <configuration>
        <annotationProcessorPaths>
            /* **原来的** */
            /* 此处添加 */
            <path>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                /* 版本与dependency一致 */
                <version>${lombok.version}</version>
            </path>
            <path>
                <groupId>org.mapstruct</groupId>
                <artifactId>mapstruct-processor</artifactId>
                <version>1.6.0.Beta2</version>
            </path>
            <path>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok-mapstruct-binding</artifactId>
                <version>0.2.0</version>
            </path>
        </annotationProcessorPaths>
    </configuration>
</plugin>

```

添加配置后，项目即可正常启动，能正常识别lombok!

Unable to render this definition异常

之所以配置实战花了2天时间，主要是因为在新项目中一点异常没有，换老项目中就报异常了，因此被这个异常弄得焦头烂额，在网上查了诸多资料，千篇一律，都是无效答案。

具体效果如下(网上盗的图):



网上的结论很多，主要分为以下几类：

1. 未指定版本(但是指定了也没用)
2. Controller路径异常，存在同类（这个编辑就无法通过）
3. Swagger注解中存在特殊字符（我压根就没用注解啊！）
4. Controller中有私有方法（纯扯蛋）

查看接口返回时发现调用api-docs接口时，返回的居然不是通用的json结构，而是一长串字符，经过一系列的排查与处理，考虑到可能是因为配置了WebMvc的extendMessageConverters导致的（stackoverflow提示我的）：

```
@Configuration
public class WebMVCConfig extends WebMvcConfigurationSupport {

    @Override
    protected void extendMessageConverters(List<HttpMessageConverter<?>> converters) {
        // 创建转换器对象
        MappingJackson2HttpMessageConverter messageConverter = new MappingJackson2HttpMessageConverter();
        // 设置类型转换器
        messageConverter.setObjectMapper(new JacksonObjectMapper());
        // 优先使用自定义转换器
        converters.addFirst(messageConverter);
    }
}
```

注释掉此段代码后，Swagger页面恢复正常！！！但是！！！这个是针对java8时间格式化的转换啊，不能注释啊！！

继续打断点！！最终发现原来Swagger也会生成一个Coverter: **Jaxb2RootElementHttpMessageConverter**，Swagger的信息转换可能强依赖这玩意，那我就只能让自定义Converter让让位完成要求：

```
@Configuration
public class WebMVCConfig implements WebMvcConfigurer {
```

```
    private static final int COVERT_PRIORITY = 6;

    @Override
    public void
extendMessageConverters(List<HttpMessageConverter<?>> converters) {
        // 创建转换器对象
        MappingJackson2HttpMessageConverter messageConverter = new
MappingJackson2HttpMessageConverter();
        // 设置类型转换器
        messageConverter.setObjectMapper(new JacksonObjectMapper());
        // 因为需要使用Springdoc以及Swagger，优先级需要慢于
Jaxb2RootElementHttpMessageConverter
        // 这个转换器是Swagger创建的，它的优先级是第5，所以需要调整自定
义Covert优先级
        // 使用自定义转换器
        converters.add(COVERT_PRIORITY, messageConverter);
    }
}

```

最后
--

至此旧有的项目完成了knife4j到Springdoc的改造调整，以后再也不用写Swagger注释了~~开森~~

希望这篇文章能给你提供些微帮助，码码快乐