链滴

SpringBoot 生成接口文档，我用 smart-doc

作者：jianzh5

原文链接：https://ld246.com/article/1632472672531

来源网站：链滴

之前我在SpringBoot老鸟系列中专门花了大量的篇幅详细介绍如何集成Swagger，以及如何对Swagger进行扩展让其支持接口参数分组功能。详情可见：SpringBoot 如何生成接口文档，老鸟们都这么玩！

可是当我接触到另一个接口文档工具 smart-doc后，我觉得它比Swagger更适合集成在项目中，更适老鸟们。今天我们就来介绍一下smart-doc组件的使用，作为对老鸟系列文章的一个补充。

swagger vs smart-doc

首先我们先看一下Swagger组件目前存在的主要问题：

1. Swagger的代码侵入性比较强

这个很容易理解，要让Swagger生成接口文档必须要给方法或字段添加对应的注解，是存在代码侵入。

2. 原生swagger不支持接口的参数分组

对于有做参数分组的接口，原生的Swagger并未支持，虽然我们通过扩展其组件可以让其支持参数分，但是毕竟要开发，而且还未支持最新的Swagger3版本。

那作为对比，smart-doc 是基于接口源码分析来生成接口文档，完全做到零注解侵入，你只需要按照java标准注释的写，smart-doc就能帮你生成一个简易明了的markdown 或是一个像GitBook样式的静html文档。官方地址：https://gitee.com/smart-doc-team/smart-doc

简单罗列一下smart-doc的优点：

- 零注解、零学习成本、只需要写标准java注释即可生成文档。
- 基于源代码接口定义自动推导，强大的返回结构推导。
- 支持Spring MVC,Spring Boot,Spring Boot Web Flux(controller书写方式)。
- 支持Callable,Future,CompletableFuture等异步接口返回的推导。
- 支持JavaBean上的JSR303参数校验规范，支持参数分组。
- 对一些常用字段定义能够生成有效的模拟值。
- ...

接下来我们来看看SpringBoot中如何集成smart-doc。

SpringBoot集成 smart-doc

smart-doc支持多种方式生成接口文档：maven插件、gradle插件、单元测试（不推荐），这里我才的是基于maven插件生成，步骤如下：

1. 引入依赖，版本选择最新版本

```
<!--引入smart-doc-->
<plugin>
  <groupId>com.github.shalousun</groupId>
  <artifactId>smart-doc-maven-plugin</artifactId>
  <version>2.2.7</version>
  <configuration>
    <configFile>./src/main/resources/smart-doc.json</configFile>
    <projectName>Smart-Doc初体验</projectName>
  </configuration>
</plugin>
```

重点在 configFile中指定smart-doc配置文件 smart-doc.json

2. 新建配置文件smart-doc.json

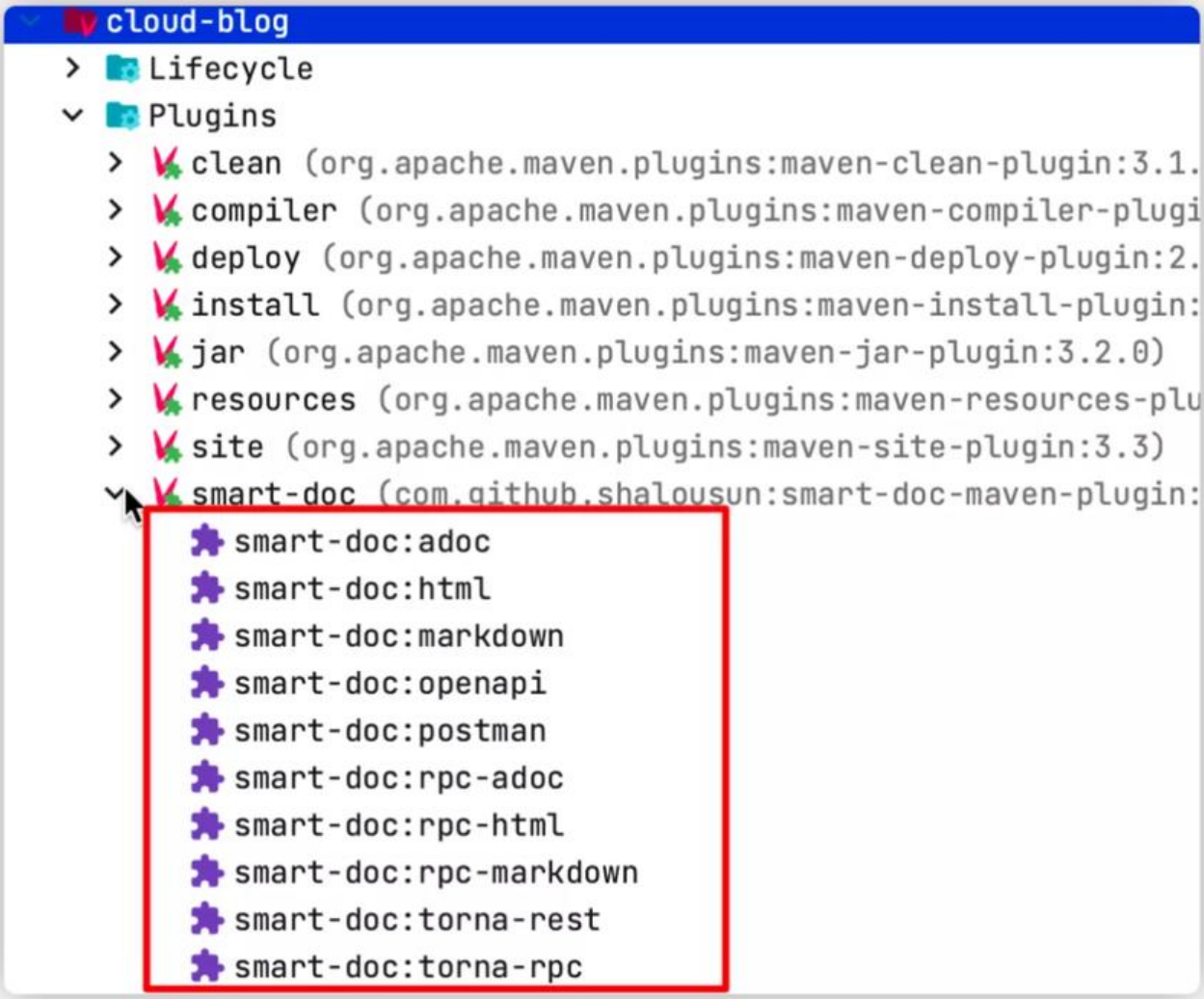
```
{
  "outPath": "src/main/resources/static/doc"
}
```

指定smart-doc生成的文档路径，其他配置项可以参考官方wiki。

3. 通过执行maven 命令生成对应的接口文档

```
//生成html
mvn -Dfile.encoding=UTF-8 smart-doc:html
```

当然也可以通过idea中的maven插件生成



4. 访问接口文档

生成接口文档后我们通过 http://localhost:8080/doc/api.html查看，效果如下：

原文链接：SpringBoot 生成接口文档，我用 smart-doc

看到这里的同学可能会呵呵一笑，就这？啥也没有呀！这还想让我替换Swagger?

别急，刚刚只是体验了smart-doc的基础功能，接下来我们通过丰富smart-doc的配置文件内容来增其功能。

功能增强

1. 开启调试

一个优秀的接口文档工具调试功能必不能少，smart-doc支持在线调试功能，只需要加入如下几个配项：

```
{
  "serverUrl": "http://localhost:8080",      -- 服务器地址
  "allInOne": true,                 -- 是否将文档合并到一个文件中，一般推荐为true
  "outPath": "src/main/resources/static/doc", -- 指定文档的输出路径
  "createDebugPage": true,           -- 开启测试
  "allInOneDocFileName":"index.html",      -- 自定义文档名称
  "projectName": "初识smart-doc"         -- 项目名称
}
```

通过"createDebugPage": true 开启debug功能，在让生成smart-doc生成文档时直接放入到 static/oc/下，这样可以直接启动程序访问页面 http://localhost:8080/doc/index.html进行开发调试。

有的开发人员直接在idea中使用【Open In Browser】打开smart-doc生成的debug页面，如果非要做，前端js请求后台接口时就出现了跨域。因此你需要在后端配置跨域。

这里以 SpringBoot2.3.x 为例配置后端跨域：

```
@Configuration
public class WebMvcAutoConfig implements WebMvcConfigurer {

    @Bean
    public CorsFilter corsFilter() {
        final UrlBasedCorsConfigurationSource urlBasedCorsConfigurationSource = new UrlBase
CorsConfigurationSource();
        final CorsConfiguration corsConfiguration = new CorsConfiguration();
        /* 是否允许请求带有验证信息 */
        corsConfiguration.setAllowCredentials(true);
        /* 允许访问的客户端域名 */
        corsConfiguration.addAllowedOrigin("*");
        /* 允许服务端访问的客户端请求头 */
        corsConfiguration.addAllowedHeader("*");
        /* 允许访问的方法名,GET POST等 */
        corsConfiguration.addAllowedMethod("*");
        urlBasedCorsConfigurationSource.registerCorsConfiguration("/**", corsConfiguration);
        return new CorsFilter(urlBasedCorsConfigurationSource);
    }
}
```

开启跨域后我们就可以直接在静态接口页面中进行调试了。

原文链接：SpringBoot 生成接口文档，我用 smart-doc

2. 通用响应体

在 “SpringBoot 如何统一后端返回格式？老鸟们都是这样玩的！” 一文中我们通过实现 ResponseBodyAdvice对所有返回值进行了包装，给前端返回统一的数据结构ResultData，我们需要让其在接口档中也有此功能，在配置文件追加配置内容：

```
{
  "responseBodyAdvice":{          -- 通用响应体
    "className":"com.jianzh5.blog.base.ResultData"
  }
}
```

Query-parameters:

	Parameter	Value	Type	Required	Description
☑	userName	jodie.watsica	string	false	用户

Response-fields:

Field	Type	Description	Since
status	int32	结果状态 ,具体状态码参见 ResultData.java	-
message	string	响应消息	-
data	object	响应数据	-
timestamp	int64	接口请求时间	-

Send Request

3. 自定义Header

在前后端分离项目中我们一般需要在请求接口时设置一个请求头，如token，Authorization等...后端据请求头判断是否为系统合法用户，目前smart-doc也对其提供了支持。

在smart-doc配置文件 smart-doc.json中继续追加如下配置内容：

```
"requestHeaders": [ //设置请求头，没有需求可以不设置
    {
      "name": "token",//请求头名称
      "type": "string",//请求头类型
      "desc": "自定义请求头 - token",//请求头描述信息
      "value":"123456",//不设置默认null
      "required": false,//是否必须
      "since": "-",//什么版本添加的改请求头
      "pathPatterns": "/smart/say",//只有以/smart/say 开头的url才会有此请求头
      "excludePathPatterns":"/smart/add,/smart/edit" // url=/app/page/将不会有该请求头
    }
]
```

效果如下：

1. Smart-Doc 功能测试

1.1. 简单Get接口

URL: http://localhost:8080/smart/say

Type: GET

Author: jam

Content-Type: application/x-www-form-urlencoded;charset=utf-8

Description: 简单Get接口

Request-headers:

Header	Value	Type	Required	Description
token	123456	string	false	自定义请求头 - token

Query-parameters:

Parameter	Value	Type	Required	Description
userName	janyce.deckow	string	false	用户

4. 参数分组

演示一下smart-doc对于参数分组的支持

1.2. 参数校验分组 - 新增

URL: http://localhost:8080/smart/add

Type: POST

Author: jam

Content-Type: application/x-www-form-urlencoded;charset=utf-8

Description: 参数校验分组 - 新增

Query-parameters:

Parameter	Value	Type	Required	Description
sex	F	string	false	性别
level	A	string	true	级别
age	30	int32	true	年龄，编辑时为空，新增时必填

原文链接：SpringBoot 生成接口文档，我用 smart-doc

新增操作时，age、level为必填项，sex为非必填。

1.3. 参数校验分组 - 编辑

URL: http://localhost:8080/smart/update

Type: PUT

Author: jam

Content-Type: application/x-www-form-urlencoded;charset=utf-8

Description: 参数校验分组 - 编辑

Query-parameters:

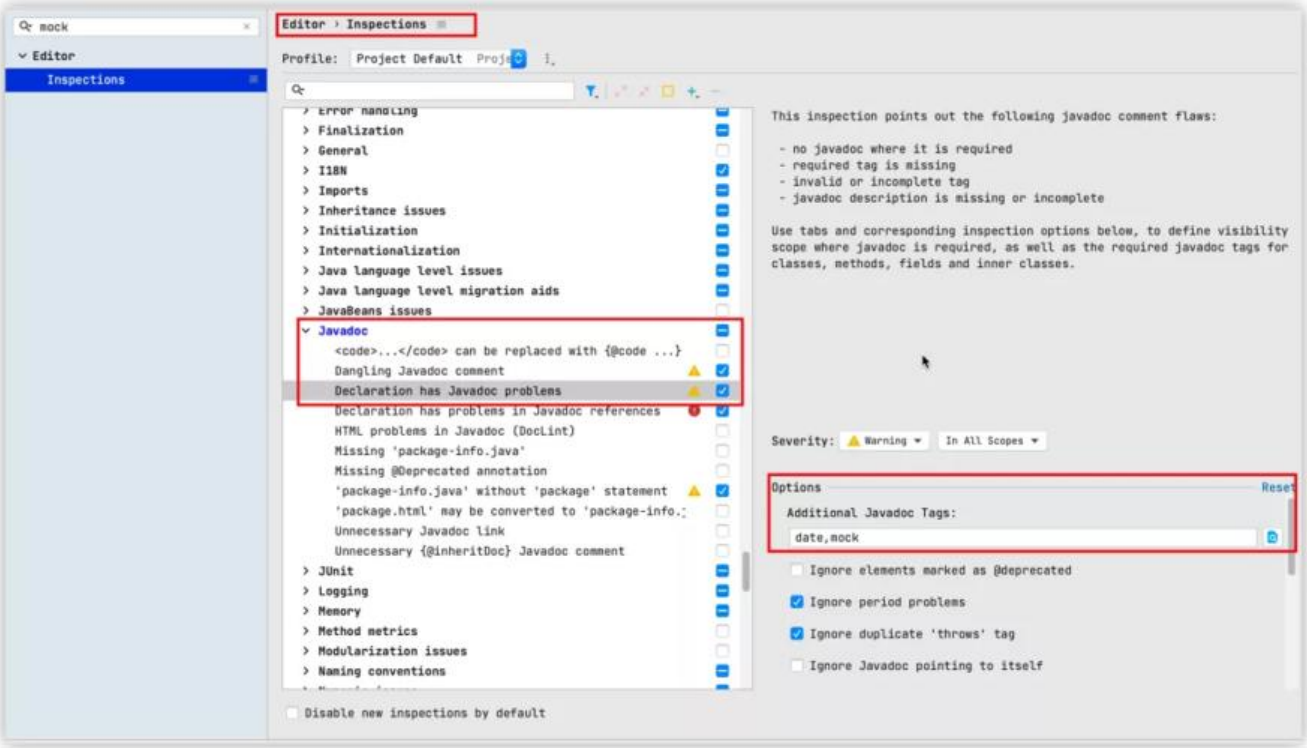
Parameter	Value	Type	Required	Description
id	190	string	true	id 新增时为空，编辑时必填
appId	190	string	true	应用ID 新增时为空，编辑时必填
sex	F	string	false	性别
level	A	string	true	级别

编辑操作时，id，appid，leven为必填项，sex为非必填。

通过上面的效果可以看出smart-doc对于参数分组是完全支持的。

5. idea配置doc

自定义的tag默认是不会自动提示的，需要用户在idea中进行设置。设置好后即可使用，下面以设置smart-doc自定义的mock tag为例，设置操作如下：



6. 完整配置

附上完整配置，如果还需要其他配置大家可以参考wiki自行引入。

```
{
  "serverUrl": "http://localhost:8080",
  "allInOne": true,
  "outPath": "src/main/resources/static/doc",
  "createDebugPage": true,
  "allInOneDocFileName":"index.html",
  "projectName": "初识smart-doc",
  "packageFilters": "com.jianzh5.blog.smartdoc.*",
  "errorCodeDictionaries": [{
    "title": "title",
    "enumClassName": "com.jianzh5.blog.base.ReturnCode",
    "codeField": "code",
    "descField": "message"
  }],
  "responseBodyAdvice":{
    "className":"com.jianzh5.blog.base.ResultData"
  },
  "requestHeaders": [{
    "name": "token",
    "type": "string",
    "desc": "自定义请求头 - token",
    "value":"123456",
    "required": false,
    "since": "-",
    "pathPatterns": "/smart/say",
    "excludePathPatterns":"/smart/add,/smart/edit"
  }]
}
```

小结

其实没什么可总结的，smart-doc使用非常简单，官方文档也非常详细，只要你会写标准的java注释可以给你生成详细的接口文档。（如果你说你不会写注释，那这篇文章可能不太适合你） 而且在引入mart-doc后还可以强制要求开发人员给接口编写注释，保证团队代码风格不会出现很大差异。

老鸟系列源码已经上传至GitHub，需要的在公众号【JAVA日知录】回复关键字 **0923** 获取