# SpringBoot2基于Swagger2生成离线Api文档

谢随安 (/u/00a60b4319f4) + 关注
2.3 2019.01.30 14:47* 字数 1258 阅读 665 评论 0 喜欢 16 阅读 665 评论 0 喜欢 16
(/u/00a60b4319f4)

> 通过swagger2与swagger-ui可以很方便的生成系统的在线api文档，这方面的博客网上有很多。
> 但是利用swagger生成离线api文档的博客就不多了。有的无法兼容springboot2，有的配置起来太麻烦，复用性与易用性较差。
> 为了能够方便的自动生成api离线文档，笔者花了些时间基于网上的博客做了修改，代码经过亲测可用于springboot2项目。
> **注意**：生成pdf离线文档时，会因为

---

## 代码样例下载地址

**Github**: https://github.com/ChaselX/springboot-swagger2-offline-api-doc (https://github.com/ChaselX/springboot-swagger2-offline-api-doc)

**Gitee**: https://gitee.com/chasel96/springboot-swagger2-offline-api-doc (https://gitee.com/chasel96/springboot-swagger2-offline-api-doc)

---

## 旧版与新版的区别

个人觉得**旧版的配置简单许多**，新版的配置按照官方demo的配置来做还是复杂了很多

> **重要提示1**： 本文档有两种版本生成，上面的只能基于2.6.1以下的swagger版本使用，下面的可以用于2.6.1以上版本。笔者在官档上看到了使用最新的swagger来生成离线api文档的方法，并已经在代码样例中实现了。
> **重要提示2**：在生成pdf离线文档时，因为adoc的字符编码问题，部分中文会消失，网上有现成的解决办法，但是由于解决的方式不够灵活，不在文档中展示。生成html离线文档不受影响。

---

# 使用2.6.1及以下版本生成Api文档

## 离线文档生成效果预览



生成文档效果预览

## 最终目标

配置到Springboot项目中以后，在项目打包的时候便会通过单元测试在指定的目录生成被官方 (https://springfox.github.io/springfox/docs/current/)称为staticdocs的离线文档

## 从Maven配置开始

### Maven依赖引入

该篇博文引用的依赖都要引入，Spring Rest Docs的依赖spring-restdocs-mockmvc，离线文档的依赖springfox-staticdocs，因为要在单元测试的时候生成文档，所以需要再加测试相关的spring-boot-starter-test。

> 注意：因为springfox-staticdocs在16年以后就没有发布新依赖包了，swagger2与swagger-ui的依赖包版本必须为2.6.1，否则会因为springfox-staticdocs的swagger-model与swagger2中的model不兼容导致错误。亲测踩坑填坑。

```
<properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEn
coding>
        <java.version>1.8</java.version>
        <snippetsDirectory>${project.build.directory}/generated-snippets</s
nippetsDirectory>
    </properties>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--swagger-->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>2.6.1</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>2.6.1</version>
        </dependency>
        <!--offline doc-->
        <dependency>
            <groupId>org.springframework.restdocs</groupId>
            <artifactId>spring-restdocs-mockmvc</artifactId>
            <version>2.0.2.RELEASE</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-staticdocs</artifactId>
            <version>2.6.1</version>
            <scope>test</scope>
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

## Maven插件

`asciidoctor-maven-plugin` 插件会把Asciidoc格式文件转成HTML5格式输出。

```
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <configuration>
                    <includes>
                        <include>**/*Documentation.java</include>
                    </includes>
                </configuration>
            </plugin>
            <plugin>
                <groupId>org.asciidoctor</groupId>
                <artifactId>asciidoctor-maven-plugin</artifactId>
                <version>1.5.3</version>
                <configuration>
                    <sourceDirectory>${project.basedir}/docs/asciidoc</sour
ceDirectory>
                    <sourceDocumentName>index.adoc</sourceDocumentName>
                    <attributes>
                        <doctype>book</doctype>
                        <toc>left</toc>
                        <toclevels>3</toclevels>
                        <numbered></numbered>
                        <hardbreaks></hardbreaks>
                        <sectlinks></sectlinks>
                        <sectanchors></sectanchors>
                        <generated>${project.build.directory}/asciidoc</gen
erated>
                    </attributes>
                </configuration>
                <!-- Since each execution can only handle one backend, run
                     separate executions for each desired output type -->
                <executions>
                    <execution>
                        <id>output-html</id>
                        <phase>test</phase>
                        <goals>
                            <goal>process-asciidoc</goal>
                        </goals>
                        <configuration>
                            <backend>html5</backend>
                            <!--<outputDirectory>${project.basedir}/docs/as
ciidoc/html</outputDirectory>-->
                        </configuration>
                    </execution>
                </executions>
                <!-- Configure generic document generation settings -->
                <dependencies>
                    <!-- add for SpringBoot 2 -->
                    <dependency>
                        <groupId>org.jruby</groupId>
                        <artifactId>jruby-complete</artifactId>
                        <version>1.7.26</version>
                    </dependency>
                </dependencies>
            </plugin>
        </plugins>
    </build>
```

# 生成文档的单元测试代码编写

```
package com.chinamobile.cmic.apidoc;

import io.github.robwin.markup.builder.MarkupLanguage;
import io.github.robwin.swagger2markup.GroupBy;
import io.github.robwin.swagger2markup.Swagger2MarkupConverter;
import org.junit.After;
import org.junit.Assert;
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.restdocs.AutoConfigureRe
stDocs;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigur
eMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.restdocs.mockmvc.MockMvcRestDocumentation;
import org.springframework.restdocs.mockmvc.RestDocumentationRequestBuilder
s;
import org.springframework.restdocs.operation.preprocess.Preprocessors;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.result.MockMvcResultMatchers;
import org.springframework.web.context.WebApplicationContext;
import springfox.documentation.staticdocs.SwaggerResultHandler;

/**
 * @author ChaselX
 * @date 2019/1/29 13:02
 */
@AutoConfigureMockMvc
@AutoConfigureRestDocs(outputDir = "target/generated-snippets")
@RunWith(SpringRunner.class)
@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM_PORT)
public class Documentation {
    private String snippetDir = "target/generated-snippets";
    private String outputDir = "target/asciidoc";

    @Autowired
    private MockMvc mockMvc;

    @Autowired
    private WebApplicationContext webApplicationContext;

    @After
    public void test() throws Exception {
        // 得到swagger.json,写入outputDir目录中
        mockMvc.perform(RestDocumentationRequestBuilders.get("/v2/api-docs"
)
                .accept(MediaType.APPLICATION_JSON))
                .andDo(SwaggerResultHandler.outputDirectory(outputDir).buil
d())
                .andExpect(MockMvcResultMatchers.status().isOk())
                .andReturn();

        // 读取上一步生成的swagger.json转成asciiDoc,写入到outputDir
        // 这个outputDir必须和插件里面<generated></generated>标签配置一致
        Swagger2MarkupConverter.from(outputDir + "/swagger.json")
                .withPathsGroupedBy(GroupBy.TAGS)// 按tag排序
                .withMarkupLanguage(MarkupLanguage.ASCIIDOC)// 格式
                .withExamples(snippetDir)
                .build()
                .intoFolder(outputDir);// 输出
```

```
    }

    @Test
    public void testApi() {
        try {
            mockMvc.perform(RestDocumentationRequestBuilders.get("/greeting
")
                    .accept(MediaType.APPLICATION_JSON))
                    .andExpect(MockMvcResultMatchers.status().isOk())
                    .andDo(MockMvcRestDocumentation.document("greeting",
                            Preprocessors.preprocessResponse(Preprocessors.
prettyPrint())))
                    .andReturn();
        } catch (Exception e) {
            e.printStackTrace();
            Assert.fail();
        }
    }
}
```

这个类包含两个方法，TestApi()是用来生成例子，test()用来生成Asciidoc的文档。生成例子用到了spring-restdocs-mockmvc，每一个API都要进行单元测试才能生成相应的文档片段（snippets），生成的结果如图：

Api用例片段

生成完整的Asciidoc文档用到了 `Swagger2MarkupConverter` ，第一步先获取在线版本的文档并保存到文件 `swagger.json` 中，第二步把 `swagger.json` 和之前的例子snippets整合并保存为Asciidoc格式的完整文档。生成结果如图：

Asciidoc格式的文档

## Swagger配置类

通过配置类定义一些文档相关的信息

```
package com.chinamobile.cmic.apidoc.config;

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

/**
 * @author ChaselX
 * @date 2019/1/30 11:22
 */
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket createRestApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                .apis(RequestHandlerSelectors.basePackage("com.chinamobile.
cmic.apidoc"))
                .paths(PathSelectors.any())
                .build();
    }

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder().title("API Document")
                .description("API Document for Spring Boot 2 Project")
                .contact(new Contact("ChaselX", "", ""))
                .version("1.0")
                .build();
    }
}
```

## index.adoc

路径：项目名/docs/asciidoc/index.adoc

```
include::{generated}/overview.adoc[]
include::{generated}/definitions.adoc[]
include::{generated}/paths.adoc[]
```

## 生成HTML5的文档

利用前面配置的maven插件，只需要执行打包就可以生成相应的文档，如图：

生成的HTML5文档

## 参考资料

1. https://www.jianshu.com/p/af7a6f29bf4f
   (https://www.jianshu.com/p/af7a6f29bf4f)
2. https://blog.csdn.net/fly910905/article/details/79131755
   (https://blog.csdn.net/fly910905/article/details/79131755)

---

# 使用2.6.1及以上版本生成Api文档

## 从Maven配置开始

### Maven依赖引入

该篇博文引用的依赖都要引入，Spring Rest Docs的依赖spring-restdocs-mockmvc，离线文档的依赖springfox-staticdocs，因为要在单元测试的时候生成文档，所以需要再加测试相关的spring-boot-starter-test。

```
   <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEn
coding>
        <java.version>1.8</java.version>

        <swagger2markup.version>1.2.0</swagger2markup.version>
        <asciidoctor.input.directory>${project.basedir}/src/docs/asciidoc</
asciidoctor.input.directory>
        <swagger.output.dir>${project.build.directory}/swagger</swagger.out
put.dir>
        <swagger.snippetOutput.dir>${project.build.directory}/asciidoc/snip
pets</swagger.snippetOutput.dir>
        <generated.asciidoc.directory>${project.build.directory}/asciidoc/g
enerated</generated.asciidoc.directory>
        <asciidoctor.html.output.directory>${project.build.directory}/ascii
doc/html</asciidoctor.html.output.directory>
        <asciidoctor.pdf.output.directory>${project.build.directory}/asciid
oc/pdf</asciidoctor.pdf.output.directory>
        <swagger.input>${swagger.output.dir}/swagger.json</swagger.input>
        <springfox.version>2.9.2</springfox.version>
```

```
    </properties>

    <repositories>
        <repository>
            <id>jcentral</id>
            <name>bintray</name>
            <url>http://jcenter.bintray.com</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </repository>
        <repository>
            <id>jcenter-snapshots</id>
            <name>jcenter</name>
            <url>http://oss.jfrog.org/artifactory/oss-snapshot-local/</url>
        </repository>
    </repositories>

    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <!--Default Log Encoder for Swagger2Markup-->
        <dependency>
            <groupId>net.logstash.logback</groupId>
            <artifactId>logstash-logback-encoder</artifactId>
            <version>5.1</version>
        </dependency>
        <!--swagger-->
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger2</artifactId>
            <version>${springfox.version}</version>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-swagger-ui</artifactId>
            <version>${springfox.version}</version>
        </dependency>
        <!--offline doc-->
        <dependency>
            <groupId>org.springframework.restdocs</groupId>
            <artifactId>spring-restdocs-mockmvc</artifactId>
            <!--<scope>test</scope>-->
        </dependency>
        <dependency>
            <groupId>io.github.swagger2markup</groupId>
            <artifactId>swagger2markup-spring-restdocs-ext</artifactId>
            <version>${swagger2markup.version}</version>
            <scope>test</scope>
        </dependency>
        <dependency>
            <groupId>io.springfox</groupId>
            <artifactId>springfox-staticdocs</artifactId>
            <version>2.6.1</version>
            <!--<scope>test</scope>-->
        </dependency>

        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
```

## Maven插件

`asciidoctor-maven-plugin` 插件会把Asciidoc格式文件转成HTML5格式输出。

```
    <pluginRepositories>
        <pluginRepository>
            <id>jcenter-snapshots</id>
            <name>jcenter</name>
            <url>http://oss.jfrog.org/artifactory/oss-snapshot-local/</url>
        </pluginRepository>
        <pluginRepository>
            <id>jcenter-releases</id>
            <name>jcenter</name>
            <url>http://jcenter.bintray.com</url>
            <snapshots>
                <enabled>false</enabled>
            </snapshots>
        </pluginRepository>
    </pluginRepositories>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-compiler-plugin</artifactId>
                <version>3.3</version>
                <configuration>
                    <compilerVersion>${java.version}</compilerVersion>
                    <source>${java.version}</source>
                    <target>${java.version}</target>
                    <encoding>UTF-8</encoding>
                    <!-- prevents endPosTable exception for maven compile -
->
                    <useIncrementalCompilation>false</useIncrementalCompila
tion>
                </configuration>
            </plugin>

            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-surefire-plugin</artifactId>
                <configuration>
                    <systemPropertyVariables>
                        <io.springfox.staticdocs.outputDir>${swagger.output
.dir}</io.springfox.staticdocs.outputDir>
                        <io.springfox.staticdocs.snippetsOutputDir>${swagge
r.snippetOutput.dir}</io.springfox.staticdocs.snippetsOutputDir>
                    </systemPropertyVariables>
                </configuration>
            </plugin>

            <!-- First, use the swagger2markup plugin to generate asciidoc
-->
            <plugin>
                <groupId>io.github.swagger2markup</groupId>
                <artifactId>swagger2markup-maven-plugin</artifactId>
                <version>${swagger2markup.version}</version>
                <dependencies>
                    <dependency>
                        <groupId>io.github.swagger2markup</groupId>
                        <artifactId>swagger2markup-import-files-ext</artifa
```

```
ctId>
                        <version>${swagger2markup.version}</version>
                    </dependency>
                    <dependency>
                        <groupId>io.github.swagger2markup</groupId>
                        <artifactId>swagger2markup-spring-restdocs-ext</art
ifactId>
                        <version>${swagger2markup.version}</version>
                    </dependency>
                </dependencies>
                <configuration>
                    <swaggerInput>${swagger.input}</swaggerInput>
                    <outputDir>${generated.asciidoc.directory}</outputDir>
                    <config>
                        <swagger2markup.markupLanguage>ASCIIDOC</swagger2ma
rkup.markupLanguage>
                        <swagger2markup.pathsGroupedBy>TAGS</swagger2markup
.pathsGroupedBy>

                        <swagger2markup.extensions.dynamicOverview.contentP
ath>${project.basedir}/src/docs/asciidoc/extensions/overview</swagger2marku
p.extensions.dynamicOverview.contentPath>
                        <swagger2markup.extensions.dynamicDefinitions.conte
ntPath>${project.basedir}/src/docs/asciidoc/extensions/definitions</swagger
2markup.extensions.dynamicDefinitions.contentPath>
                        <swagger2markup.extensions.dynamicPaths.contentPath
>${project.basedir}/src/docs/asciidoc/extensions/paths</swagger2markup.exte
nsions.dynamicPaths.contentPath>
                        <swagger2markup.extensions.dynamicSecurity.contentP
ath>${project.basedir}src/docs/asciidoc/extensions/security/</swagger2marku
p.extensions.dynamicSecurity.contentPath>

                        <swagger2markup.extensions.springRestDocs.snippetBa
seUri>${swagger.snippetOutput.dir}</swagger2markup.extensions.springRestDoc
s.snippetBaseUri>
                        <swagger2markup.extensions.springRestDocs.defaultSn
ippets>true</swagger2markup.extensions.springRestDocs.defaultSnippets>
                    </config>
                </configuration>
                <executions>
                    <execution>
                        <phase>test</phase>
                        <goals>
                            <goal>convertSwagger2markup</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>

            <!-- Run the generated asciidoc through Asciidoctor to generate
                 other documentation types, such as PDFs or HTML5 -->
            <plugin>
                <groupId>org.asciidoctor</groupId>
                <artifactId>asciidoctor-maven-plugin</artifactId>
                <version>1.5.6</version>
                <!-- Include Asciidoctor PDF for pdf generation -->
                <dependencies>
                    <dependency>
                        <groupId>org.asciidoctor</groupId>
                        <artifactId>asciidoctorj-pdf</artifactId>
                        <version>1.5.0-alpha.16</version>
                    </dependency>
                    <dependency>
                        <groupId>org.jruby</groupId>
                        <artifactId>jruby-complete</artifactId>
                        <version>1.7.21</version>
                    </dependency>
```

```
                </dependencies>
                <!-- Configure generic document generation settings -->
                <configuration>
                    <sourceDirectory>${asciidoctor.input.directory}</source
Directory>
                    <sourceDocumentName>index.adoc</sourceDocumentName>
                    <attributes>
                        <doctype>book</doctype>
                        <toc>left</toc>
                        <toclevels>3</toclevels>
                        <numbered></numbered>
                        <hardbreaks></hardbreaks>
                        <sectlinks></sectlinks>
                        <sectanchors></sectanchors>
                        <generated>${generated.asciidoc.directory}</generat
ed>
                    </attributes>
                </configuration>
                <!-- Since each execution can only handle one backend, run
                     separate executions for each desired output type -->
                <executions>
                    <execution>
                        <id>output-html</id>
                        <phase>test</phase>
                        <goals>
                            <goal>process-asciidoc</goal>
                        </goals>
                        <configuration>
                            <backend>html5</backend>
                            <outputDirectory>${asciidoctor.html.output.dire
ctory}</outputDirectory>
                        </configuration>
                    </execution>

                    <execution>
                        <id>output-pdf</id>
                        <phase>test</phase>
                        <goals>
                            <goal>process-asciidoc</goal>
                        </goals>
                        <configuration>
                            <backend>pdf</backend>
                            <outputDirectory>${asciidoctor.pdf.output.direc
tory}</outputDirectory>
                        </configuration>
                    </execution>

                </executions>
            </plugin>

            <!-- specify the main class for the manifest -->
            <plugin>
                <groupId>org.apache.maven.plugins</groupId>
                <artifactId>maven-jar-plugin</artifactId>
                <version>3.1.0</version>
                <configuration>
                    <archive>
                        <manifest>
                            <addClasspath>true</addClasspath>
                            <classpathPrefix>lib/</classpathPrefix>
                            <!--important!!! specify the main class for the
manifest!!!-->
                            <!--important!!! specify the main class for the
manifest!!!-->
                            <!--important!!! specify the main class for the
manifest!!!-->
                            <mainClass>com.chinamobile.cmic.apidoc.ApiDocAp
```

```
plication</mainClass>
                        </manifest>
                    </archive>
                </configuration>
            </plugin>

            <!-- copy dependencies to the lib directory -->
            <plugin>
                <artifactId>maven-dependency-plugin</artifactId>
                <executions>
                    <execution>
                        <phase>package</phase>
                        <goals>
                            <goal>copy-dependencies</goal>
                        </goals>
                        <configuration>
                            <outputDirectory>${project.build.directory}/lib
</outputDirectory>
                        </configuration>
                    </execution>
                </executions>
            </plugin>

            <!-- copy the generated documents -->
            <plugin>
                <artifactId>maven-resources-plugin</artifactId>
                <version>3.1.0</version>
                <executions>
                    <execution>
                        <id>copy-resources</id>
                        <phase>prepare-package</phase>
                        <goals>
                            <goal>copy-resources</goal>
                        </goals>
                        <configuration>
                            <outputDirectory>${project.build.outputDirector
y}/static/docs</outputDirectory>
                            <resources>
                                <resource>
                                    <directory>${asciidoctor.html.output.di
rectory}</directory>
                                </resource>
                                <resource>
                                    <directory>${asciidoctor.pdf.output.dir
ectory}</directory>
                                </resource>
                            </resources>
                        </configuration>
                    </execution>
                </executions>
            </plugin>
        </plugins>
    </build>
```

## 生成文档的单元测试代码编写

```
package com.chinamobile.cmic.apidoc;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.autoconfigure.restdocs.AutoConfigureRe
stDocs;
import org.springframework.boot.test.autoconfigure.web.servlet.AutoConfigur
```

```
eMockMvc;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.http.MediaType;
import org.springframework.mock.web.MockHttpServletResponse;
import org.springframework.test.context.junit4.SpringRunner;
import org.springframework.test.context.web.WebAppConfiguration;
import org.springframework.test.web.servlet.MockMvc;
import org.springframework.test.web.servlet.MvcResult;

import java.io.BufferedWriter;
import java.nio.charset.StandardCharsets;
import java.nio.file.Files;
import java.nio.file.Paths;

import static org.springframework.test.web.servlet.request.MockMvcRequestBu
ilders.get;
import static org.springframework.test.web.servlet.result.MockMvcResultMatc
hers.status;

/**
 * @author ChaselX
 * @date 2019/1/29 13:02
 */
@WebAppConfiguration
@RunWith(SpringRunner.class)
@AutoConfigureRestDocs(outputDir = "build/asciidoc/snippets")
@SpringBootTest
@AutoConfigureMockMvc
public class Swagger2MarkupTest {

    @Autowired
    private MockMvc mockMvc;

    @Test
    public void testApi() throws Exception {
        mockMvc.perform(get("/greeting")
                .accept(MediaType.APPLICATION_JSON))
                .andDo(document("greetingGet",
                        Preprocessors.preprocessResponse(Preprocessors.pret
tyPrint())))
                .andExpect(status().isOk());
    }

    @Test
    public void createSpringfoxSwaggerJson() throws Exception {

        String outputDir = System.getProperty("io.springfox.staticdocs.outp
utDir");
        MvcResult mvcResult = this.mockMvc.perform(get("/v2/api-docs")
                .accept(MediaType.APPLICATION_JSON))
                .andExpect(status().isOk())
                .andReturn();

        MockHttpServletResponse response = mvcResult.getResponse();
        String swaggerJson = response.getContentAsString();
        Files.createDirectories(Paths.get(outputDir));
        try (BufferedWriter writer = Files.newBufferedWriter(Paths.get(outp
utDir, "swagger.json"), StandardCharsets.UTF_8)) {
            writer.write(swaggerJson);
        }
    }
}
```

这个类包含两个方法，TestApi()是用来生成例子，createSpringfoxSwaggerJson()用来生成Asciidoc的文档。生成例子用到了spring-restdocs-mockmvc，每一个API都要进行单元测试才能生成相应的文档片段（snippets），生成的结果如图：

Api用例片段

生成完整的Asciidoc文档用到了 `Swagger2MarkupConverter` ，第一步先获取在线版本的文档并保存到文件 `swagger.json` 中，第二步把 `swagger.json` 和之前的例子snippets整合并保存为Asciidoc格式的完整文档。生成结果如图：

生成结果

## Swagger配置类

通过配置类定义一些文档相关的信息

```
package com.chinamobile.cmic.apidoc.config;

import com.google.common.base.Predicates;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import springfox.documentation.builders.ApiInfoBuilder;
import springfox.documentation.service.ApiInfo;
import springfox.documentation.service.Contact;
import springfox.documentation.spi.DocumentationType;
import springfox.documentation.spring.web.plugins.Docket;
import springfox.documentation.swagger2.annotations.EnableSwagger2;

import static springfox.documentation.builders.PathSelectors.ant;

/**
 * @author ChaselX
 * @date 2019/1/30 11:22
 */
@Configuration
@EnableSwagger2
public class SwaggerConfig {

    @Bean
    public Docket restApi() {
        return new Docket(DocumentationType.SWAGGER_2)
                .apiInfo(apiInfo())
                .select()
                .paths(Predicates.and(ant("/**"), Predicates.not(ant("/erro
r"))))
                .build();
    }

    private ApiInfo apiInfo() {
        return new ApiInfoBuilder()
                .title("API Document")
                .description("API Document for Spring Boot 2 Project")
                .contact(new Contact("ChaselX", "", ""))
                .version("1.0")
                .build();
    }
}
```

## LogstashEncoder

在resources目录下创建一个名为logback.xml的配置文件，使用LogstashEncoder作为Default Log Encoder

```
<?xml version="1.0" encoding="UTF-8"?>
<configuration scan="true">
    <conversionRule conversionWord="clr" converterClass="org.springframewor
k.boot.logging.logback.ColorConverter" />
    <conversionRule conversionWord="wex" converterClass="org.springframewor
k.boot.logging.logback.WhitespaceThrowableProxyConverter" />

    <property name="CONSOLE_LOG_PATTERN" value="%clr(%d{yyyy-MM-dd HH:mm:ss
.SSS}){faint} %clr(%5p) %clr(${PID:- }){magenta} %clr(---){faint} %clr([%15
.15t{14}]){faint} %clr(%-40.40logger{39}){cyan} %clr(:){faint} %m%n%wex"/>

    <springProperty scope="context" name="application_name" source="info.na
me"/>
    <springProperty scope="context" name="application_version" source="info
.version"/>
    <springProperty scope="context" name="environment" source="info.environ
ment"/>

    <jmxConfigurator/>

    <appender name="CONSOLE" class="ch.qos.logback.core.ConsoleAppender">
        <encoder class="net.logstash.logback.encoder.LogstashEncoder" />
    </appender>

    <root level="INFO">
        <appender-ref ref="CONSOLE" />
    </root>

</configuration>
```

## index.adoc

路径：项目名src/docs/asciidoc/index.adoc

```
include::{generated}/overview.adoc[]
include::{generated}/paths.adoc[]
include::{generated}/security.adoc[]
include::{generated}/definitions.adoc[]
```

## 生成HTML5、PDF的文档

利用前面配置的maven插件，只需要执行打包就可以生成相应的文档，如图：

生成的HTML5文档

生成的pdf文档

**小礼物走一走，来简书关注我**

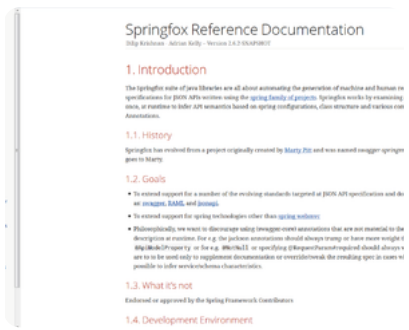
赞赏支持

被以下专题收入，发现更多相似内容

Spring ... (/c/04217c089f34?utm_source=desktop&utm_medium=notes-included-collection)

技术方案 (/c/d8036c9aef89?utm_source=desktop&utm_medium=notes-included-collection)

(/p/af7a6f29bf4f?




utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend

**SpringBoot项目生成RESTfull API的文档 (/p/af7a6f29bf4f?utm_campai...**

本人所在的项目团队分为前端开发和后端开发两个子小组，前后端通过RESTfull API通信，过去一般都要用word来写API文档，随着需求的变化和开发的深入，往往还要多次更新API文档，这会给开发人员增加不少

quiterr (/u/e268fe04200a?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

---

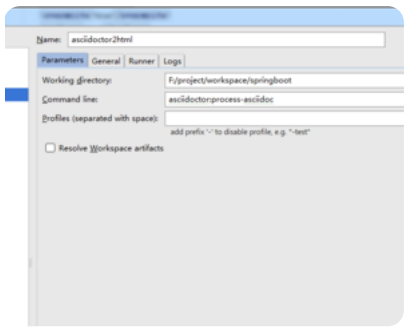
**基于Spring的Restful接口生成工具 (/p/ecb8daa4ecf7?utm_campaign=...**

场景 有时候需要为前端开发者提供Restful Api说明文档，通过word文档创建和修改非常耗时，希望有一种比较便捷的第三方库可以减少生成Api说明文档的工作量 基于Spring的Restful Api生成工具 术语解析

飞天豌豆狼 (/u/3de004df8054?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

---

(/p/f0b1ed00c411?



utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend

**使用Swagger2Markup实现导出API文档 (/p/f0b1ed00c411?utm_campai...**

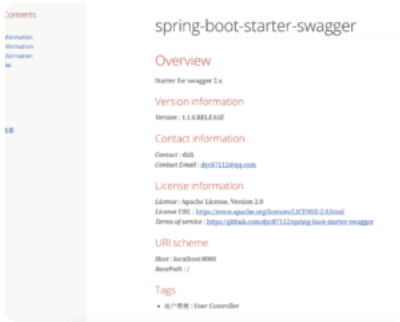
前言 在学会了如何使用Swagger之后，我们已经能够轻松地为Spring MVC或SpringBoot的Web项目自动构建出API文档了。但是，构建的文档必须通过在项目中整合swagger-ui、或使用单独部署的swagger-ui

binnan (/u/e73a02fe13e2?

utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

---

(/p/087e17015fb0?




utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend

**使用Swagger2Markup实现API文档的静态部署（一）： AsciiDoc (/p/08...**

在阅读本文之前，您先需要了解Swagger的使用，如果您还不知道它是用来干嘛的，请先阅读《Spring Boot中使用Swagger2构建强大的RESTful API文档》一文。 前言 在学会了如何使用Swagger之后，我们已

程序猿DD (/u/6a622d516e32?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

---

(/p/6eaa6182abed?


utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend

**在Spring REST API中使用Swagger2进行文档管理 (/p/6eaa6182abed?u...**

写在前面 使用RESTful API作为Web服务对外提供服务的入口，基本上已经成为了标准，在提供REST API的同时，如何进行API文档管理是一个较为麻烦的事情，作为开发人员我们都了解API文档的重要性，但总

程序员精进 (/u/21cf3250e09a?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

---

(/p/82f7c311e40f?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend

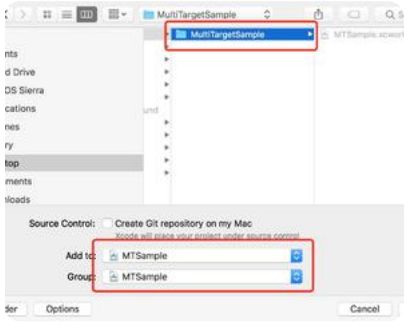
**生气到底有多可怕? 医生说这些病都是气出来的! (/p/82f7c311e40f?utm_...**

我们嘴边常挂一句话“气死我了”。“气死”不是夸张，生活中真有人被气死。容易生气、愤怒，不仅坏了心情，还会引发很多健康问题。 俗话说，“不生气就不生病”。生气就像是一场“大地震”，气在心情，伤在身

沃享财富 (/u/28164616ac81?
utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommend

---

(/p/818aed25b8e9?




utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommend

**Workspace + Framework + MultiTarget + Cocoapods... (/p/818aed25b...**

简介 本文主要介绍这样的一种项目结构：整个工程以一个Workspace来管理，这个Workspace中包含一到多个Project，一个Project中包含一到多个Target，每个Target可以输出一个静态库（Framework）或者

乌鸢 (/u/91e6f8976ea4?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

---

## 第一次去老婆家 (/p/24635bbc1662?utm_campaign=maleskine&utm_c...

第一次去老婆家，她家住二十楼！ 正巧那天电梯维修，没办法爬楼梯上去。 开门的是她妈妈，我刚开口喊了句：“阿姨好！” 双腿瞬间一软“扑通”一下跪在她妈面前。 我尴尬的站起来准备解释！ 他爸在旁边调侃

我只是来讲个笑话 (/u/41ffdf52c150?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

---

(/p/13dce70251ca?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommenc

## 迟来的春天。(/p/13dce70251ca?utm_campaign=maleskine&utm_cont...

曾经听到一个故事，情节老套，似乎在我们的日常里，司空见惯。 阿东最近在装修自己的房子。哥们九子见了，问他，咋，准备换换风格啊？阿东停下来，转过头的时候，脸上堆满幸福。 这是我和小梅要住的新

二货笨笨 (/u/d75f4eded695?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc

---

(/p/77190051a770?

utm_campaign=maleskine&utm_content=note&utm_medium=seo_notes&utm_source=recommenc

## 50平没有风格可言，业主的要求就是不拘泥于风格 (/p/77190051a770?ut...

试试搜相似的模型、案例 客户是一个文艺气息浓郁的单身男士，对家的要求是舒适 随性 安静，不拘泥于风格。好的作品是设计师和业主一起完成的，足够信任和支持，很庆幸本篇的业主是这样的人，所以后面

无言的设计 (/u/b4a761b2e81c?utm_campaign=maleskine&utm_content=user&utm_medium=seo_notes&utm_source=recommenc