# 极速开发框架SpringBoot及源码分析

主讲：Cat 老师

**北京动力节点教育科技有限公司**  
**2020 • 北京**

**动力节点版权所有，禁止私自传播**

## 快速回顾

What is SpringBoot？

SpringBoot开发运行环境要求

springboot程序的几种创建方式

SpringBoot开发java程序

SpringBoot开发web程序

SpringBoot main方法分析

SpringBoot项目约定的代码结构

SpringBoot 项目pom.xml继承结构

SpringBoot 项目pom.xml修改默认版本

SpringBoot 整合mybatis

SpringBoot 整合jsp视图展示（也支持一些模板技术）

SpringBoot 项目pom.xml热部署插件devtools

JRebel （这是idea插件，收费的，需要破解）

SpringBoot开发最佳实践

SpringBoot敏感配置信息加密

SpringBoot数据库连接池

SpringBoot跨域资源共享CORS

SpringBoot静态资源处理

SpringBoot核心配置文件

## SpringBoot核心配置文件

1、.properties文件

键值对的properties属性文件配置方式

SpringBoot默认读取该文件作为项目配置文件

2、.yml文件

yml 也是一种配置文件格式，主要采用空格、换行、冒号等格式排版进行配置；

yml 后缀也可以使用 .yaml 后缀；

配置的值与前面的冒号必须要有一个空格；

只要在类路径上具有SnakeYAML的jar包，SpringApplication类就会自动支持YAML作为配置文件，默认的spring-boot-starter自动依赖了SnakeYAML；

具体使用可以参考一下视频演示；

3、多环境配置文件

#比如配置测试环境

spring.profiles.active=dev

application-dev.properties

#比如配置生产环境

spring.profiles.active=product

application-product.properties

3、配置文件随机值

配置随机值，通过配置文件得到一个随机值，springboot提供RandomValuePropertySource可用于注入随机值（例如，注入一些随机值或测试用例），它可以产生整数，longs，uuid或字符串，如下示例所示：

my.secret=${random.value}  
my.number=${random.int}  
my.bignumber=${random.long}  
my.uuid=${random.uuid}  
my.number.less.than.ten=${random.int(10)}  
my.number.in.range=${random.int[1024,65535]}

4、配置文件中可以使用占位符：

my.name = cat  
my.desc = ${my.name} is a teacher.

5、Duration单位转换：

30秒超时：

30，PT30S、30s都是等效的；

500ms超时：

500，PT0.5S和500ms都是等效的；

也可以使用任何受支持的单位，包括：

ns纳秒

us 微秒

ms 毫秒（毫秒）

s秒

m分钟

H 小时

d天

w周

m月

y年

比如：

server.servlet.session.timeout=1800s

server.servlet.session.timeout=30m

server.servlet.session.timeout=PT30m

6、DataSize 类型单位转换

若要指定10 MB的缓冲区大小，则10和10MB是等效的；

256个字节的大小阈值可以指定为256或256B；

B for bytes

KB for kilobytes

MB for megabytes

GB for gigabytes

TB for terabytes

比如：server.tomcat.max-swallow-size=2048KB

## application.properties/yml文件读取

1、通过[@Value](https://my.oschina.net/u/3193899)注解读取；（配置项比较少的时候）

2、通过映射类读取；（配置项目比较多的时候）

新建一个配置映射类，只需要说明配置的前缀

@ConfigurationProperties(prefix = "myconfig.user")//前缀

public class User {

private String name;//自动匹配name

private int sex;//自动匹配sex

private int age;//自动匹配age

}

3、通过Environment对象读取；（spring ioc直接就给我们创建了该环境对象）

从环境对象中读取配置；

@Autowired

Environment env;

@GetMapping("/user")

Public String user() {

StringBuilder sb = new StringBuilder();

sb.append("从Environment中获取配置：\n");

sb.append(env.getProperty("myconfig.user.name"));//注意这里不加${}

return sb.toString();

}

4、通过嵌套静态内部类读取（参考server.前缀的方式）

## SpringBoot线程池异步调用

SpringBoot框架自动装配提供了一个的线程池，用于提交异步任务；

有了线程池我们就只需要写任务即可，提交任务采用@Async注解；（底层是aop）

Springboot自动装配提供的线程池还可以根据需要进行配置：

*#SpringBoot配置线程池,修改线程池默认配置*spring.task.execution.pool.max-size=99999  
spring.task.execution.thread-name-prefix=cat-task-  
spring.task.execution.pool.queue-capacity=99999

我们也可以覆盖SpringBoot自动配置的线程池，用我们自定义的线程池：

@Bean  
public ThreadPoolTaskExecutor asyncExecutor() {  
 *logger*.info("start asyncExecutor......");  
 ThreadPoolTaskExecutor executor = new ThreadPoolTaskExecutor();  
 *//配置核心线程数* executor.setCorePoolSize(16);  
 *//配置最大线程数* executor.setMaxPoolSize(64);  
 *//配置队列大小* executor.setQueueCapacity(9999);  
 *//配置线程池中的线程的名称前缀* executor.setThreadNamePrefix("async-order-");  
 *// rejection-policy：当pool已经达到max pool size的时候，如何处理新任务  
 // CALLER\_RUNS：不在新线程中执行任务，而是由调用线程（提交任务的线程）处理该任务* executor.setRejectedExecutionHandler(new ThreadPoolExecutor.CallerRunsPolicy());  
 *//执行初始化* executor.initialize();  
 return executor;  
}

## SpringBoot集成Swagger

说到Swagger（丝袜哥），首先了解一下OpenAPI规范（OpenAPI Specification 简称OAS）是Linux基金会的一个项目，试图通过定义一种用来描述API格式或API定义的语言，来规范RESTful服务开发过程，目前版本是V3.0，并且已经发布并开源在github上。（https://github.com/OAI/OpenAPI-Specification）；

Swagger是目前最受欢迎的OpenAPI规范（OAS）开发工具框架，支持从设计和文档到测试和部署的整个API生命周期的开发；

wagger官网：https://swagger.io/ 是一个开源项目，也就是提供jar包)；

Swagger2 版本：1.x，2.x，现在都用2.x；

Spring Boot也集成了Swagger，可以很方便地在springboot中使用Swagger生成api接口文档；

**Swagger的作用：**

随项目自动生成强大RESTful API文档，减少开发人员工作量；（不需要自己写api接口文档了），使用swagger，只需要在代码中添加一些注解即可生成API接口文档，便于同步更新API文档的说明，当然也有人诟病说这种方式与代码耦合在一起，智者见智仁者见仁；

另外生成的api文档页面带有测试功能，可以用来调试每个RESTful API接口；

**怎么集成？**

springboot集成第三方组件一般情况下是三个步骤：

1. 加依赖（可能是starter）；
2. 配置application.properties文件；
3. 写代码；

所以：

1. 添加相关依赖

<!-- springfox-swagger2 -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.9.2</version>

</dependency>

<!-- springfox-swagger-ui -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.9.2</version>

</dependency>

1. 配置文件（暂时不需要在application.properties配置）
2. 编写相关代码，需要用到一些注解；

@Api(**"springboot使用swagger测试"**)  
@RestController  
**public class** SwaggerController {  
  
 @ApiOperation(value = **"获取用户信息"**, notes = **"根据id来获取用户详细信息"**)  
 @ApiImplicitParams({  
 @ApiImplicitParam(paramType = **"path"**, name = **"id"**, value = **"用户ID"**, dataType = **"Integer"**),  
 @ApiImplicitParam(paramType = **"path"**, name = **"status"**, value = **"用户状态"**, dataType = **"Integer"**)  
 })  
 @ApiResponses({  
 @ApiResponse(code = 400, message = **"缺少必要的请求参数"**),  
 @ApiResponse(code = 404, message = **"请求路径没有或页面跳转路径不对"**)  
 })

@RequestMapping(value = **"/swagger/{id}/{status}"**,method = RequestMethod.***GET***)  
**public** Users getInfo(@PathVariable(**"id"**) Integer id, @PathVariable(**"status"**) Integer status) {  
 Users users = **new** Users();  
 users.setNick(**"cat"**);  
 users.setPhone(**"1370000000"**);  
 users.setPassword(**"\*\*\*\*\*\*"**);  
 users.setEmail(**"cat@163.com"**);  
 users.setAccount(**"NO68958886878664"**);  
 **return** users;  
 }  
}

标红色的这些都是swagger的注解；

然后在spring ioc容器中配置创建一个Docket对象；

@EnableSwagger2 *//开启Swagger2的支持*@Configuration *//配置类，== applicationContext.xml*public class SwaggerConfig {  
  
 */\*\*  
 \* 在spring容器配置一个bean对象  
 \*  
 \* @Bean等价于 <bean id="createRestApi" class="xxxx.Docket">  
 \*  
 \* @return  
 \*/* @Bean  
 public Docket docket() {  
 *//链式编程（构建器模式），基本是固定；* return new Docket(DocumentationType.*SWAGGER\_2*)  
 .enable(enable)*//可以开启或关闭swagger文档* .apiInfo(apiInfo())  
 .select()  
 .apis(RequestHandlerSelectors.*withMethodAnnotation*(ApiOperation.class))*//扫描有ApiOperation注解的方法* .paths(PathSelectors.*any*())  
 .build();  
 }  
  
 */\*\*  
 \* 创建api的基本信息  
 \*  
 \* @return  
 \*/* private ApiInfo apiInfo() {  
 *//链式编程（构建器模式），基本是固定；* return new ApiInfoBuilder()  
 .title("用户账号中心接口文档")  
 .description("集成Swagger2构建RESTful APIs")  
 .termsOfServiceUrl("http://www.bjpowernode.com/")  
 .contact(new Contact("cat","http://www.bjpowernode.com","cat@163.com"))  
 .license("采用 Apache 2.0 开源许可证")  
 .licenseUrl("http://http://www.bjpowernode.com/licence.txt")  
 .version("1.0.0")  
 .build();  
 }  
}

具体参考课堂上的样例代码；

完成后访问：<http://localhost:8080/swagger-ui.html> 得到API文档；（如果有项目上下文地址要带上项目上下文）

说明：[2.10.5](https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui/2.10.5) 这个版本目前使用的时候有点问题，后续再看一下；

接口api的数据查看：[http://localhost:8080/24-springboot/v2/api-docs](http://localhost:8080/24-springboot/v2/api-docs" \t "http://localhost:8080/24-springboot/swagger-ui.html" \l "/swagger-controller/_blank)

**Swagger常用注解：**

@Api：用在类上，说明该类的作用；

@ApiOperation：用在方法上，说明方法的作用；

@ApiImplicitParams：用在方法上包含一组参数说明；

@ApiImplicitParam：用在@ApiImplicitParams注解中，指定一个请求参数的各个方面:

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-->请求参数的获取：@PathVariable （用于restful接口）；*

*body（不常用）；*

*form（不常用）；*

name： 参数名；

dataType：参数类型；

required：参数是否必须传；

value：参数的意思；

defaultValue：参数的默认值；

@ApiResponses：用于表示一组响应；

@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 "请求参数不合法"；*

*response：抛出异常的类；*

@ApiIgnore：使用该注解忽略这个API，在页面上不显示；

@ApiModel：描述一个Model的信息；

@ApiModelProperty：描述一个model的属性；

注解可参考官方：<https://github.com/swagger-api/swagger-core/wiki/Annotations>

WebJars是将Web前端Javascript和CSS等资源打包成Java的Jar包,这样在Java Web开发中我们可以借助Maven这些依赖库的管理,保证这些Web资源版本唯一性；

swagger安全性问题: 正式上线的时候，建议关闭swagger；

在配置文件配置一个开关：开启或关闭：

swagger.enable=true/false

@Value("${swagger.enable}")  
private Boolean enable;

*//链式编程（构建器模式），基本是固定；*return new Docket(DocumentationType.*SWAGGER\_2*)  
 .enable(enable) //开启或关闭  
 .apiInfo(apiInfo())  
 .select()  
 .apis(RequestHandlerSelectors.*withMethodAnnotation*(ApiOperation. class))*//扫描有ApiOperation注解的方法* .paths(PathSelectors.*any*())  
 .build();

## springfox与swagger的关系

swagger是一个流行的API开发框架，这个框架实现了“开放API声明”（OpenAPI Specification，OAS）规范；

由于Spring的流行，Marty Pitt编写了一个基于Spring的组件swagger-springmvc，用于将swagger集成到springmvc中来，而springfox则是从swagger-springmvc这个组件发展而来，同时springfox也是一个新的项目，目前新的项目中都是使用springfox-swagger2，其实在一些比较老的项目中可能还会用swagger-springmvc；

springfox-swagger2依然是实现OSA规范，生成一个描述API的json文件，，另外一个组件springfox-swagger-ui就是将这个json文件解析出来，用一种更友好的页面效果方式呈现出来；

## Lombok介绍

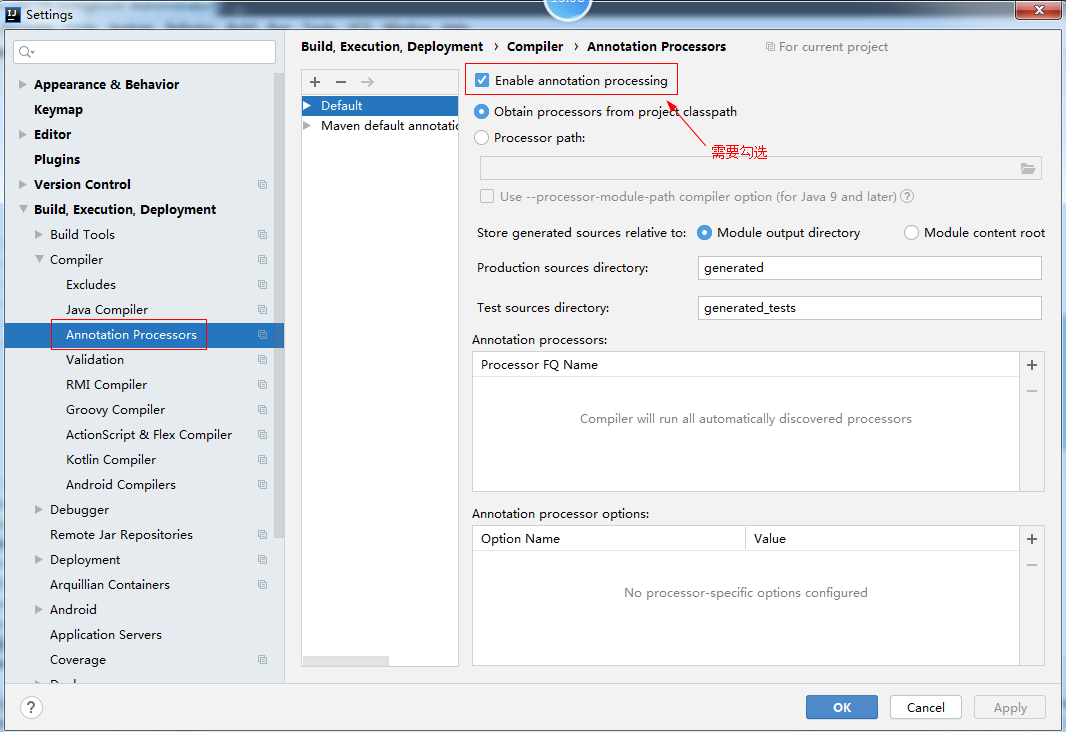
lombok就是一个jar包，加到项目中；

使用lombok的步骤如下：

1、加入lombok的依赖；

*<!--lombok-->*<dependency>  
 <groupId>org.projectlombok</groupId>  
 <artifactId>lombok</artifactId>  
 <version>1.18.12</version>  
</dependency>

2、Idea工具需要改一个配置；



3、写代码的时候，在代码上加一些lombok的注解；常用的注解：

@NoArgsConstructor  
@Getter  
@Setter  
@Data