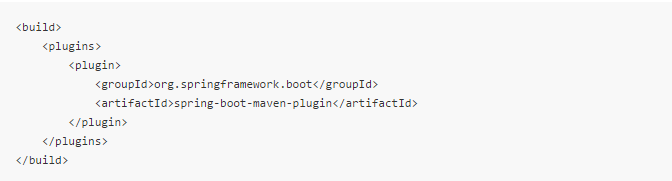
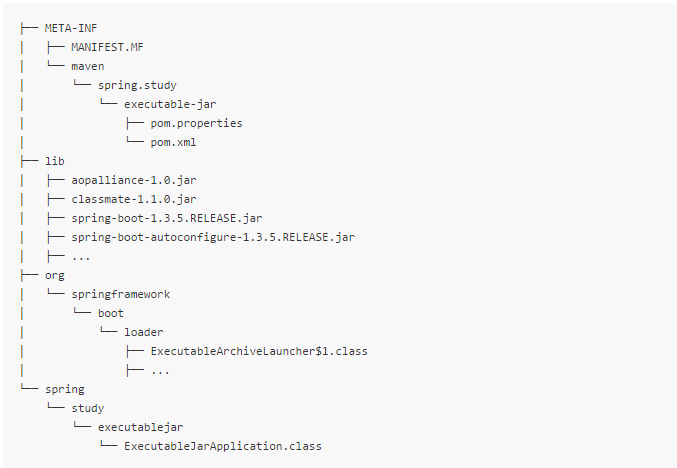
https://mp.weixin.qq.com/s/23sz3QSYiTjKgCOqGVVyxg

# 为什么SpringBoot的 jar 可以直接运行？

SpringBoot提供了一个插件spring-boot-maven-plugin用于把程序打包成一个可执行的jar包。在pom文件里加入这个插件即可：



打包完生成的executable-jar-1.0-SNAPSHOT.jar内部的结构如下：



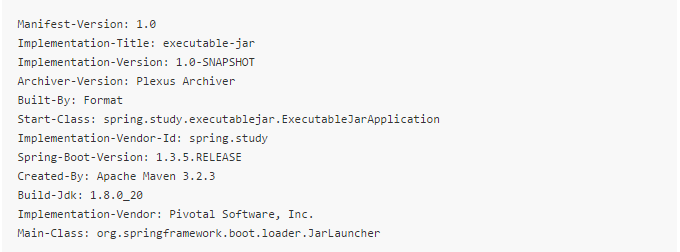
然后可以直接执行jar包就能启动程序了：

java -jar executable-jar-1.0-SNAPSHOT.jar

打包出来fat jar内部有4种文件类型：

* META-INF文件夹：程序入口，其中MANIFEST.MF用于描述jar包的信息
* lib目录：放置第三方依赖的jar包，比如springboot的一些jar包
* spring boot loader相关的代码
* 模块自身的代码

MANIFEST.MF文件的内容：



我们看到，它的Main-Class是**org.springframework.boot.loader.JarLauncher**，当我们使用java -jar执行jar包的时候会调用JarLauncher的main方法，而不是我们编写的SpringApplication。

那么JarLauncher这个类是的作用是什么的？

它是SpringBoot内部提供的工具Spring Boot Loader提供的一个用于执行Application类的工具类(fat jar内部有spring loader相关的代码就是因为这里用到了)。相当于Spring Boot Loader提供了一套标准用于执行SpringBoot打包出来的jar。

### Spring Boot Loader抽象的一些类

**抽象类Launcher**：各种Launcher的基础抽象类，用于启动应用程序；跟Archive配合使用；目前有3种实现，分别是JarLauncher、WarLauncher以及PropertiesLauncher；

**Archive**：归档文件的基础抽象类。JarFileArchive就是jar包文件的抽象。它提供了一些方法比如getUrl会返回这个Archive对应的URL；getManifest方法会获得Manifest数据等。ExplodedArchive是文件目录的抽象

**JarFile**：对jar包的封装，每个JarFileArchive都会对应一个JarFile。JarFile被构造的时候会解析内部结构，去获取jar包里的各个文件或文件夹，这些文件或文件夹会被封装到Entry中，也存储在JarFileArchive中。如果Entry是个jar，会解析成JarFileArchive。

比如一个JarFileArchive对应的URL为：

jar:file:/Users/format/Develop/gitrepository/springboot-analysis/springboot-executable-jar/target/executable-jar-1.0-SNAPSHOT.jar!/

它对应的JarFile为：

/Users/format/Develop/gitrepository/springboot-analysis/springboot-executable-jar/target/executable-jar-1.0-SNAPSHOT.jar

这个JarFile有很多Entry，比如：



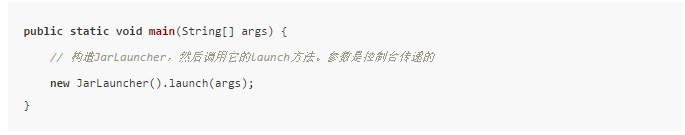
JarFileArchive内部的一些依赖jar对应的URL(SpringBoot使用org.springframework.boot.loader.jar.Handler处理器来处理这些URL)：

jar:file:/Users/Format/Develop/gitrepository/springboot-analysis/springboot-executable-jar/target/executable-jar-1.0-SNAPSHOT.jar!/lib/spring-boot-starter-web-1.3.5.RELEASE.jar!/  
  
jar:file:/Users/Format/Develop/gitrepository/springboot-analysis/springboot-executable-jar/target/executable-jar-1.0-SNAPSHOT.jar!/lib/spring-boot-loader-1.3.5.RELEASE.jar!/org/springframework/boot/loader/JarLauncher.class

我们看到如果有jar包中包含jar，或者jar包中包含jar包里面的class文件，那么会使 用 !/ 分隔开，这种方式只有org.springframework.boot.loader.jar.Handler能处理，它是SpringBoot内部扩展出来的一种URL协议。

### JarLauncher的执行过程

JarLauncher的main方法：

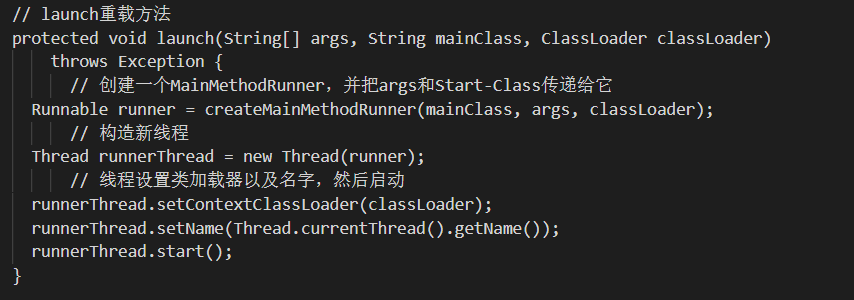


JarLauncher被构造的时候会调用父类ExecutableArchiveLauncher的构造方法。

ExecutableArchiveLauncher的构造方法内部会去构造Archive，这里构造了JarFileArchive。构造JarFileArchive的过程中还会构造很多东西，比如JarFile，Entry …

JarLauncher的launch方法：





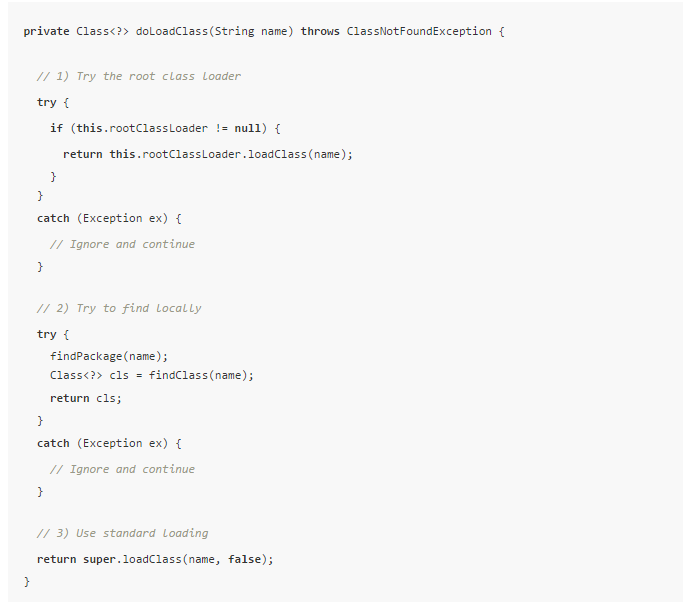
MainMethodRunner的run方法：



Start-Class的main方法调用之后，内部会构造Spring容器，启动内置Servlet容器等过程。这些过程我们都已经分析过了。

### 关于自定义的类加载器LaunchedURLClassLoader

LaunchedURLClassLoader重写了loadClass方法，也就是说它修改了默认的类加载方式(先看该类是否已加载这部分不变，后面真正去加载类的规则改变了，不再是直接从父类加载器中去加载)。LaunchedURLClassLoader定义了自己的类加载规则：



加载规则：

* 如果根类加载器存在，调用它的加载方法。这里是根类加载是ExtClassLoader
* 调用LaunchedURLClassLoader自身的findClass方法，也就是URLClassLoader的findClass方法
* 调用父类的loadClass方法，也就是执行默认的类加载顺序(从BootstrapClassLoader开始从下往下寻找)

LaunchedURLClassLoader自身的findClass方法：



下面是LaunchedURLClassLoader的一个测试：



### Spring Boot Loader的作用

SpringBoot在可执行jar包中定义了自己的一套规则，比如第三方依赖jar包在/lib目录下，jar包的URL路径使用自定义的规则并且这个规则需要使用org.springframework.boot.loader.jar.Handler处理器处理。它的Main-Class使用JarLauncher，如果是war包，使用WarLauncher执行。这些Launcher内部都会另起一个线程启动自定义的SpringApplication类。

这些特性通过spring-boot-maven-plugin插件打包完成。