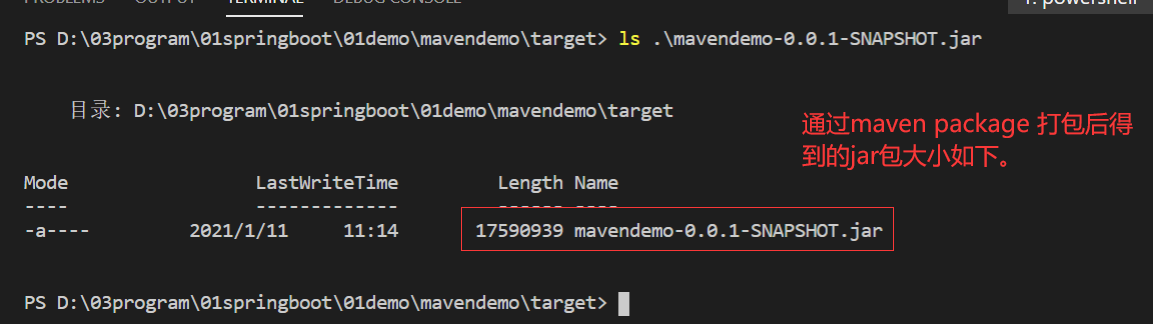
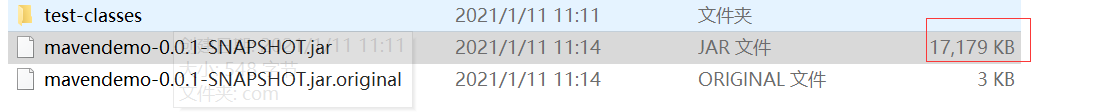
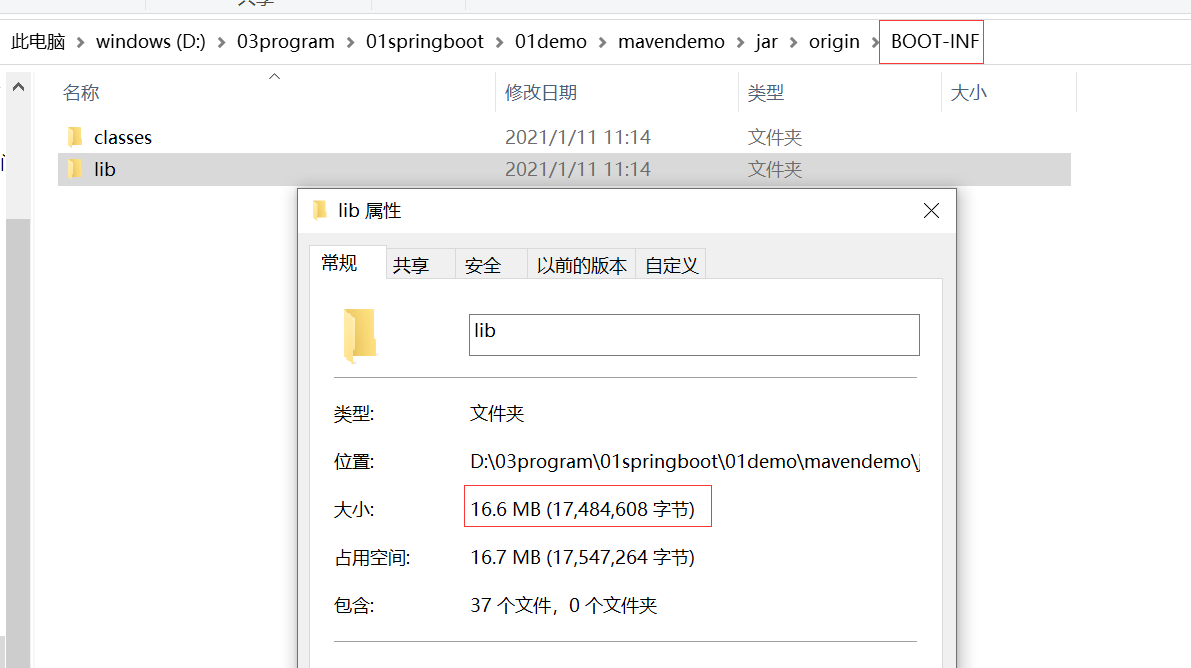
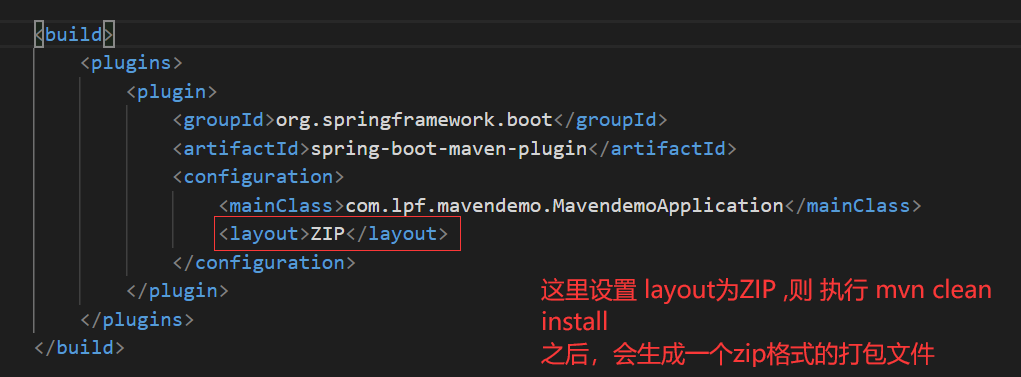
# SpringBoot瘦身

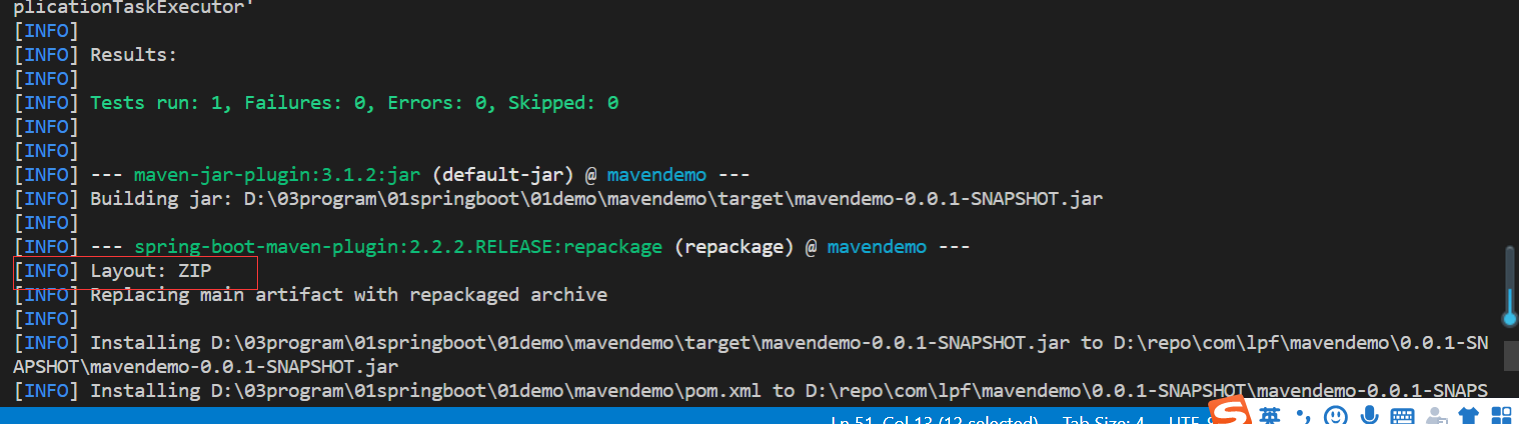


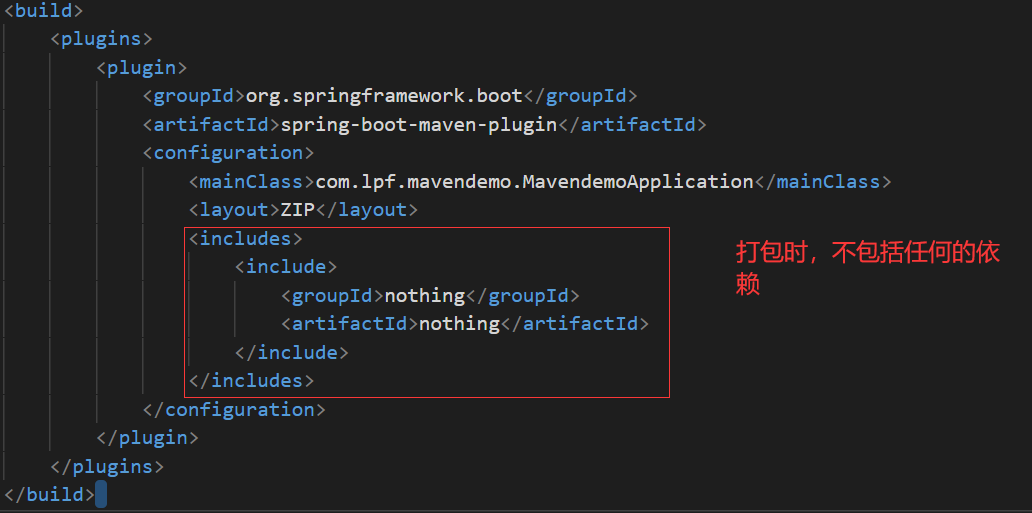


大小是 17179 kb。









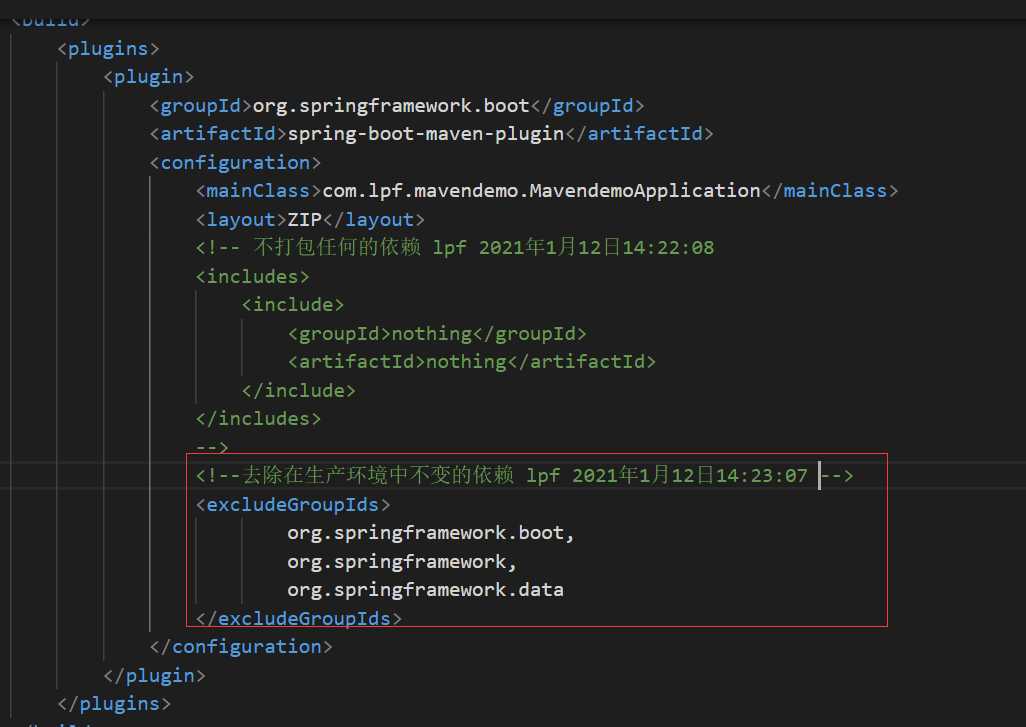
配置信息如下：

|  |
| --- |
| <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <configuration>  <mainClass>com.lpf.mavendemo.MavendemoApplication</mainClass>  <layout>ZIP</layout>  <includes>  <include>  <groupId>nothing</groupId>  <artifactId>nothing</artifactId>  </include>  </includes>  </configuration>  </plugin>  </plugins>  </build> |

上面的配置是说，打包的可执行jar包不包含任何的依赖。

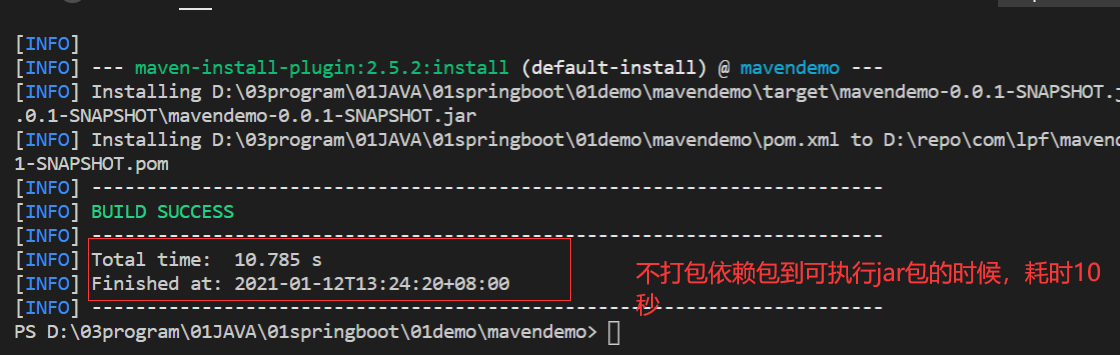
除此之外，还有另一种配置方式：就是将指定的依赖排除在可执行jar包之外。

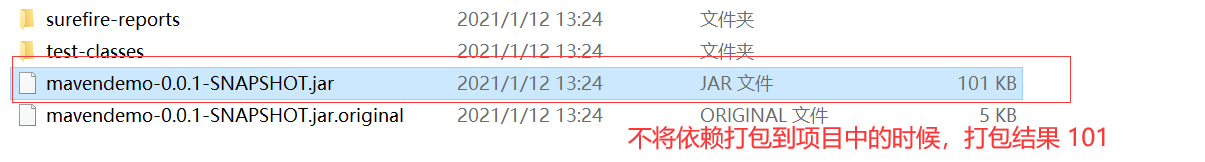
配置内容如下：



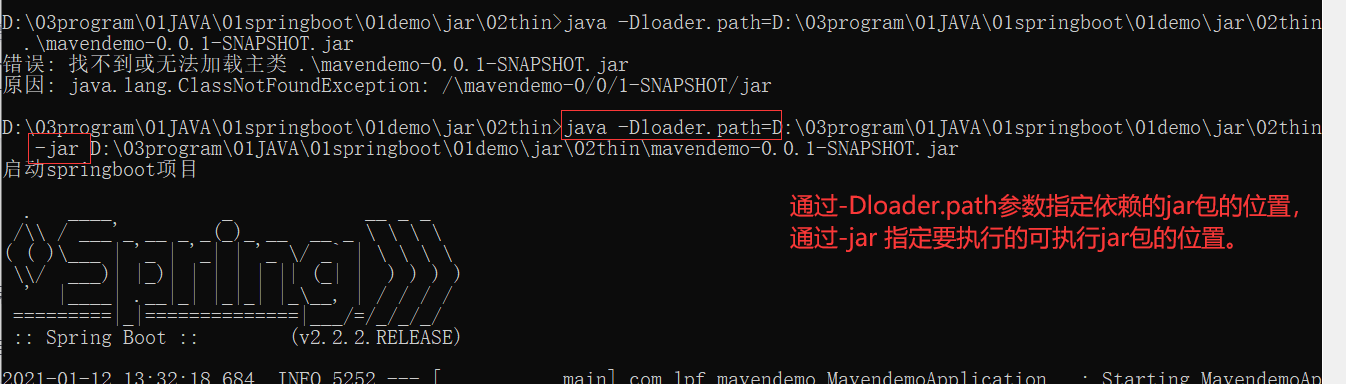
|  |
| --- |
| <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  <configuration>  <mainClass>com.lpf.mavendemo.MavendemoApplication</mainClass>  <layout>ZIP</layout>  <!-- 不打包任何的依赖 lpf 2021年1月12日14:22:08  <includes>  <include>  <groupId>nothing</groupId>  <artifactId>nothing</artifactId>  </include>  </includes>  -->  <!--去除在生产环境中不变的依赖 lpf 2021年1月12日14:23:07 -->  <excludeGroupIds>  org.springframework.boot,  org.springframework,  org.springframework.data  </excludeGroupIds>  </configuration>  </plugin>  </plugins>  </build> |

这里需要注意： 通过excludeGroupIds 这种方法会将依赖分为两类，一类在可执行jar包之外，一类在可执行jar包之内。并且同一个springboot-starter 中的jar包都不在一起。容易造成混乱。 不建议采用。



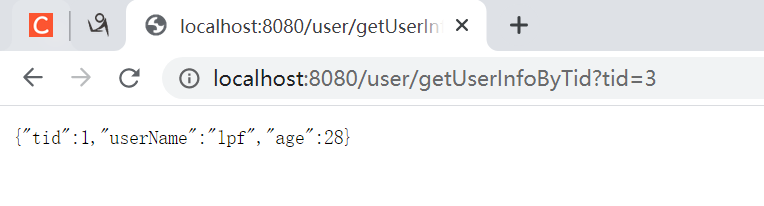


运行jar 包 ： java 命令通过-Dloader.path 和 –jar 这两个参数指定依赖包的路径和 可执行jar 包的路径。



从图中，可以看到springboot项目将lib依赖和代码分离后依然可以运行。

启动后，项目可以成功启动。



结论： 通过打包的时候，将依赖排除在可执行jar包之外，可以很好的给可执行jar包瘦身，但是对打包耗时，没有多大的影响(通过下面的耗时比较得知)

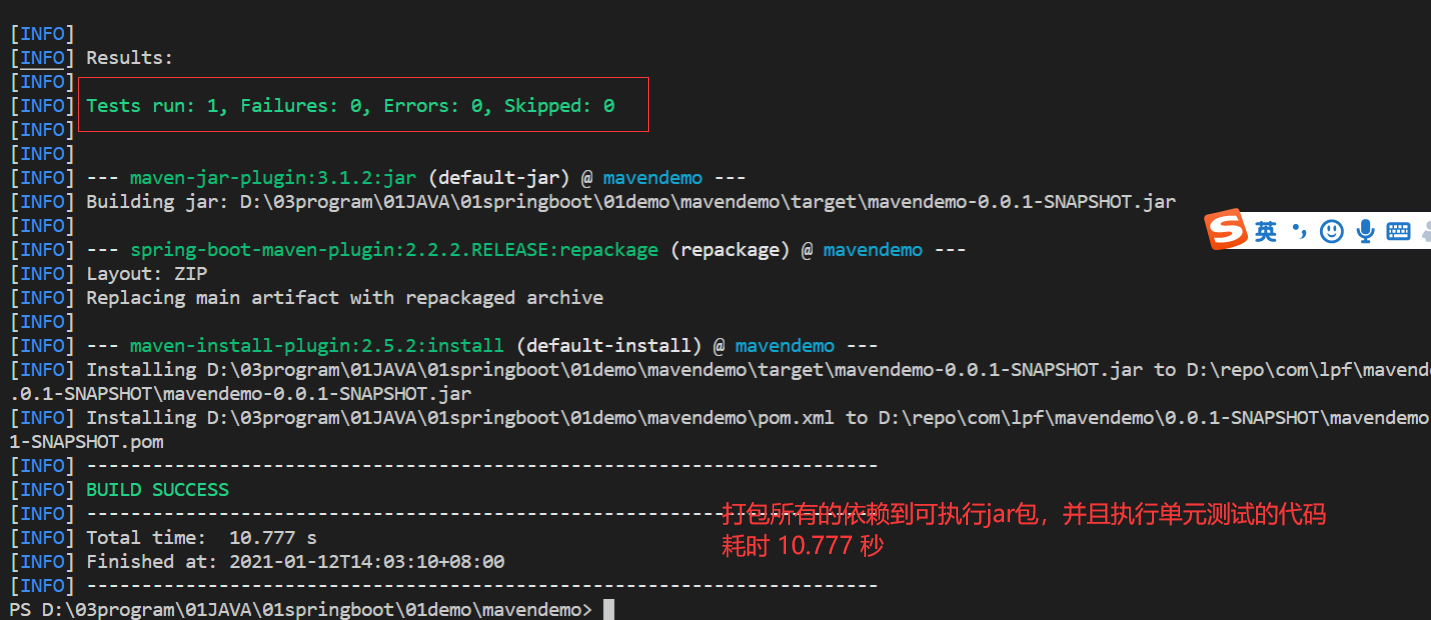
### 耗时比较

#### 场景一

将所有依赖包到打包到可执行jar包，并且执行单元测试。

耗时： 10秒多

相关命令： mvn clean install

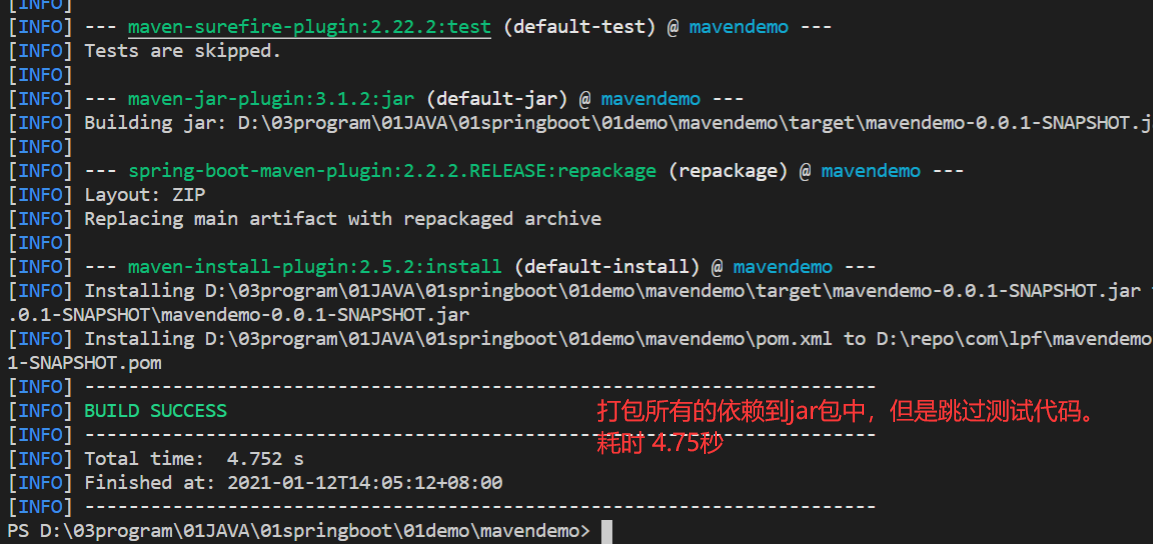


#### 场景二

将所有依赖都打包到可执行jar 包，但是，打包过程中，不进行单元测试。

耗时：5秒左右

相关命令： mvn clean install -DskipTests

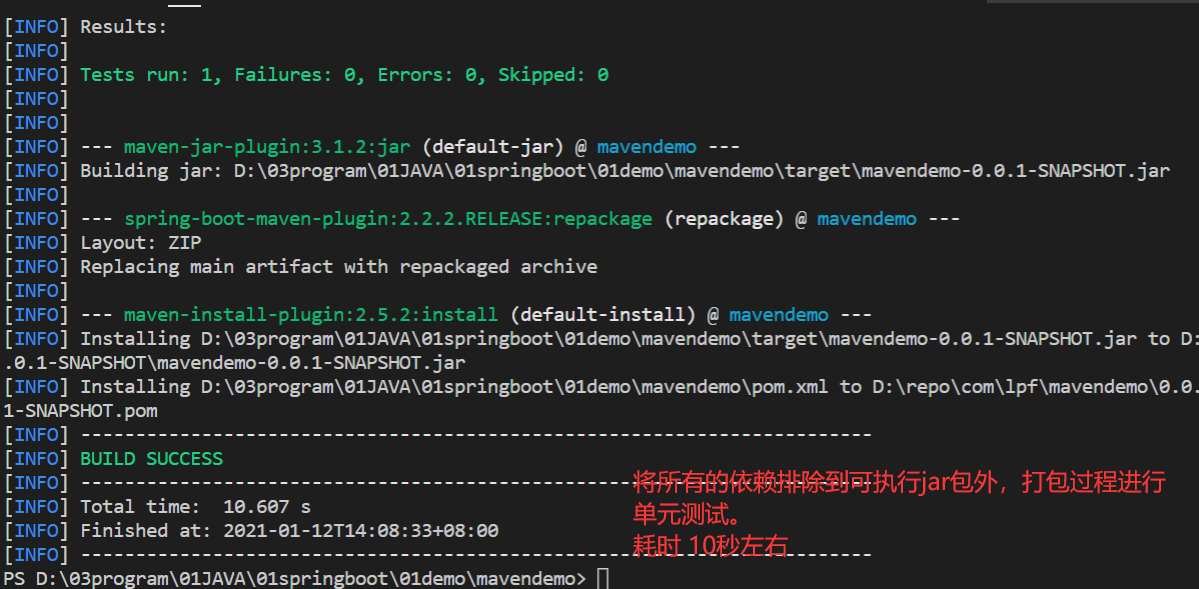


#### 场景三

将所有的依赖都排除可执行jar中，但是执行打包过程中执行单元测试，

耗时10秒多。

相关命令： mvn clean install



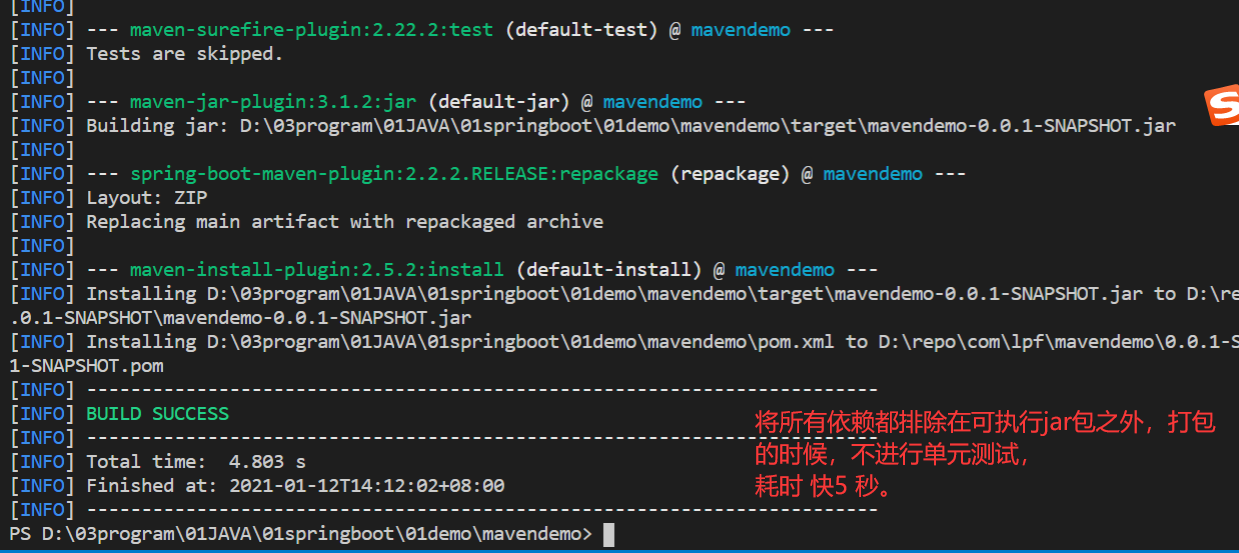
不将依赖排除在可执行jar 包之外，并且执行单元测试的时候，得到的可执行jar中，包含测试代码编译得到的class 文件。

#### 场景四

将所有的依赖都排斥在可执行jar包之外，并且打包过程中，不执行单元测试

耗时 秒。

相关命令： mvn clean install -DskipTests



打包得到的可执行jar包中，不包含测试代码。

#### 结论

通过上面的比较，可以知道是否将依赖打包到可执行jar包，对打包执行的时间没有多大的影响，真正影响打包时间的时候，在打包过程中，是否执行测试。

这次测试的时候，所依赖的jar比较少，同时，相关的java 业务代码也不多。 结果不一定准确。需要用大数据量来进行比较。

## 参考资料：

<https://www.cnblogs.com/yueli/p/7810796.html> (网页的字体很好看)

<https://blog.csdn.net/yjgithub/article/details/80475521>