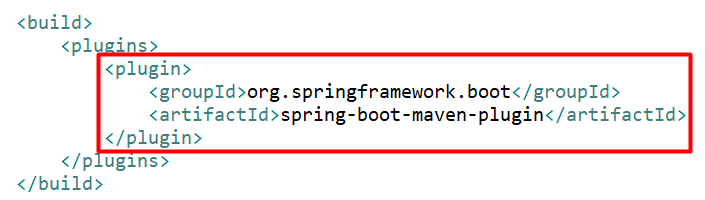
# 简介

Spring Boot项目可以内嵌Servlet容器，因此它的部署变得极为方便，可以直接打成可执行的JAR包部署在有Java运行环境（JRE）的服务器上，也可以像传统的Java Web应用程序那样打成war包部署在tomcat服务器上。

# 可执行的JAR包

1. 使用spring-boot-maven-plugin插件可以创建一个可执行的JAR应用程序（前提是应用程序的parent为spring-boot-starter-parent），所以需要配置spring-booot-maven-plugin插件，如下：

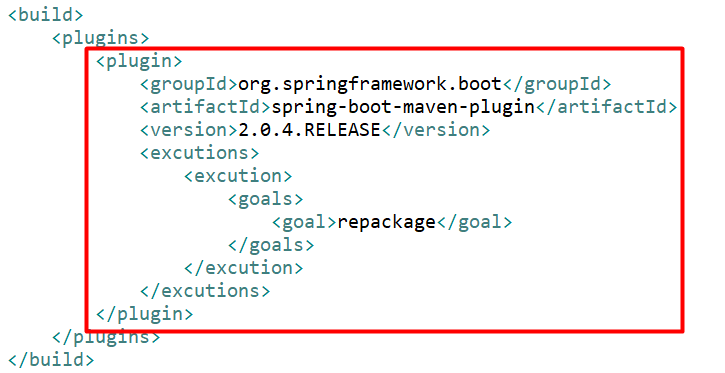


1. 修改打包方式，修改为jar，如下：



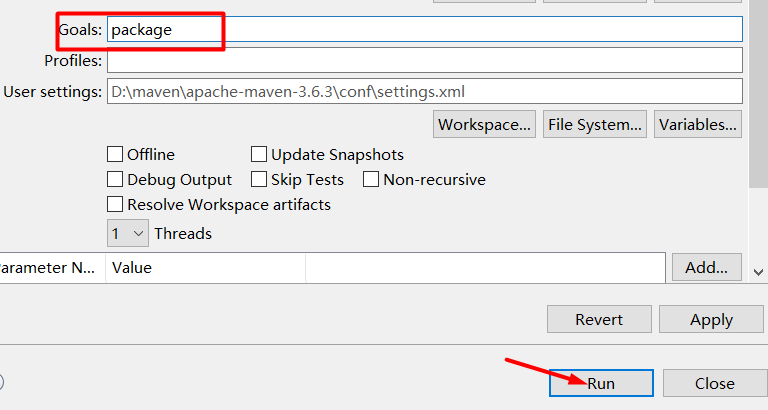
如果在pom.xml文件中不存在<packing>，则可以不用修改，因为默认打包方式就是jar包；如果修改后项目上出现了红叉，记得更新一下项目；

1. 如果项目不使用spring-boot-starter-parent，而是使用公司提供的parent，则需要对spring-boot-maven-plugin插件做一些配置，如下：



关于不使用spring-boot-stater-parent，请参考第二章第一节；

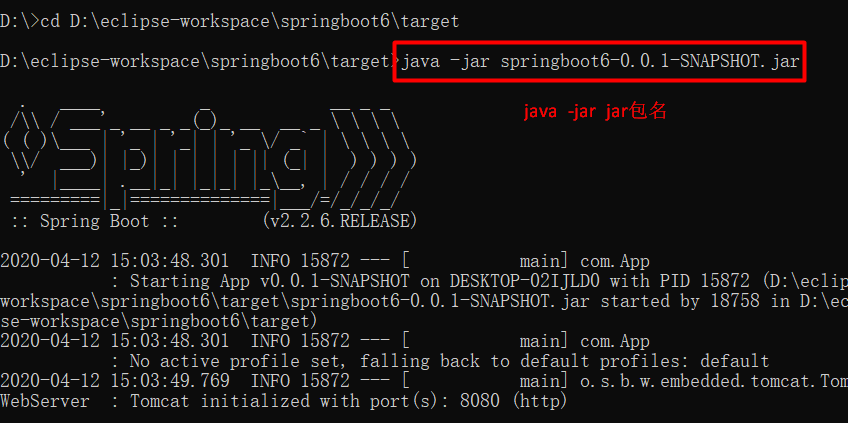
4）使用Maven命令对项目进行打包，如下：



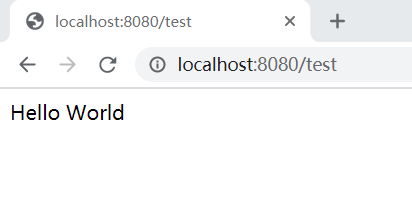
5）打包完成后，会在项目的target目录生成一个jar文件，如下：



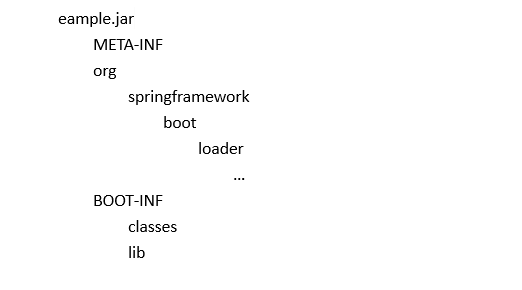
6）进入jar文件的存放路径，执行以下命令启动项目，如下：



7）之后，即可访问，如下：



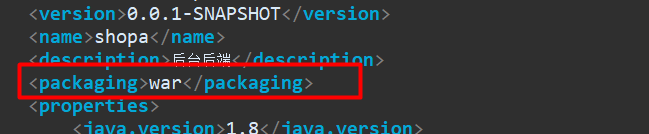
8）可执行jar包的结构图如下：



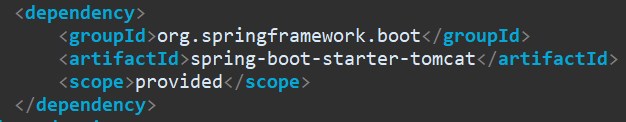
可以自行打开jar包观察内部结构；

# 可部署的WAR包

1. 修改pom.xml文件，表示将项目打成war包，如下：

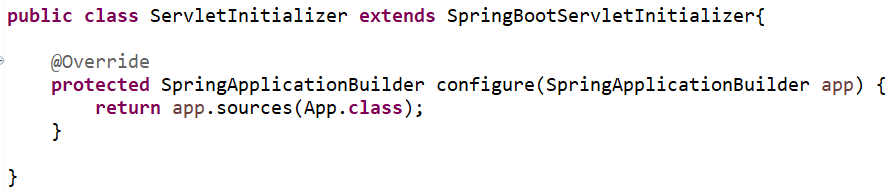


1. 修改pom.xml文件，将内嵌容器的依赖作用域修改为provide，代码如下：



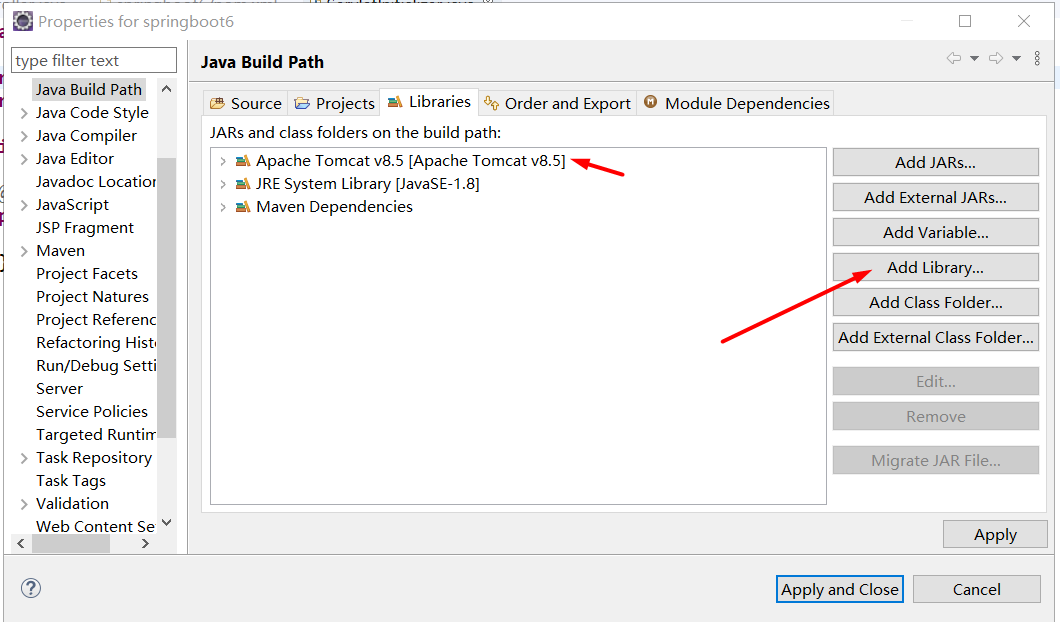
表示内嵌容器只在编译和测试时有效，运行、打包时无效（请参考Maven依赖作用域）。

1. 提供一个SpringBootServletInitializer的子类，并覆写configure方法，完成初始化操作，如下：



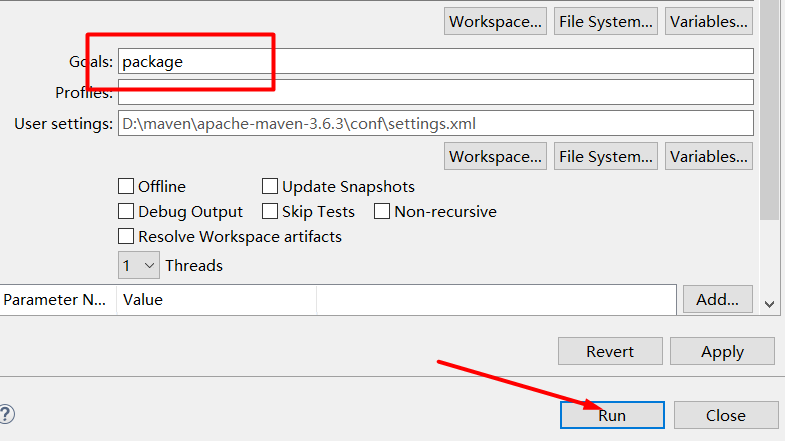
注意：App.class表示的是项目的启动类；

1. 如果编写完SpringBootServletInitializer的子类报错了，此时需要给项目添加一个服务器，选中项目右击找到Properties，然后选择添加一个服务器，如下：

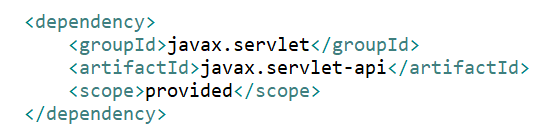


注意：记得更新一下Maven项目；

1. 使用maven命令执行打包，如下：



1. 如果在打包时报错“找不到javax.servlet.ServletException”，则在pom.xml文件中添加依赖，如下：



1. 打包成功后，会在target目录出现一个war包，



1. 启动tomcat服务器，进入tomcat管理页，将此包部署在tomcat服务器上，完成项目的部署。
2. 部署完成后，即可访问应用程序。如下：

