# #Java 命令执行jar 传参

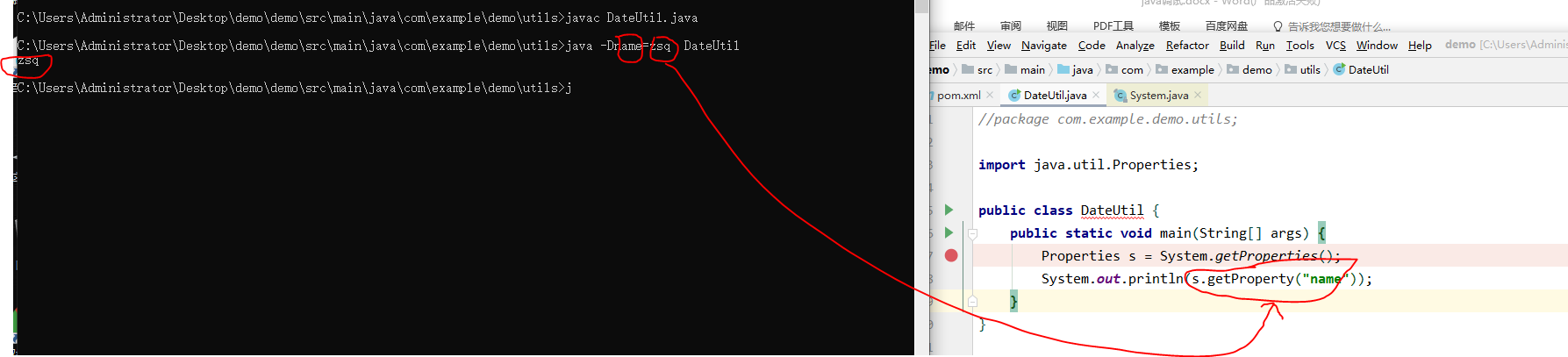
1、传递系统参数

-D<name>=<value> -D作为前缀，设置系统属性。

等价于：setProperty**(**String key, String value**)**，其目的是为了在代码运行前配置一些系统属性。

常见用法：

(1)java -D<name>=<value> javaClass

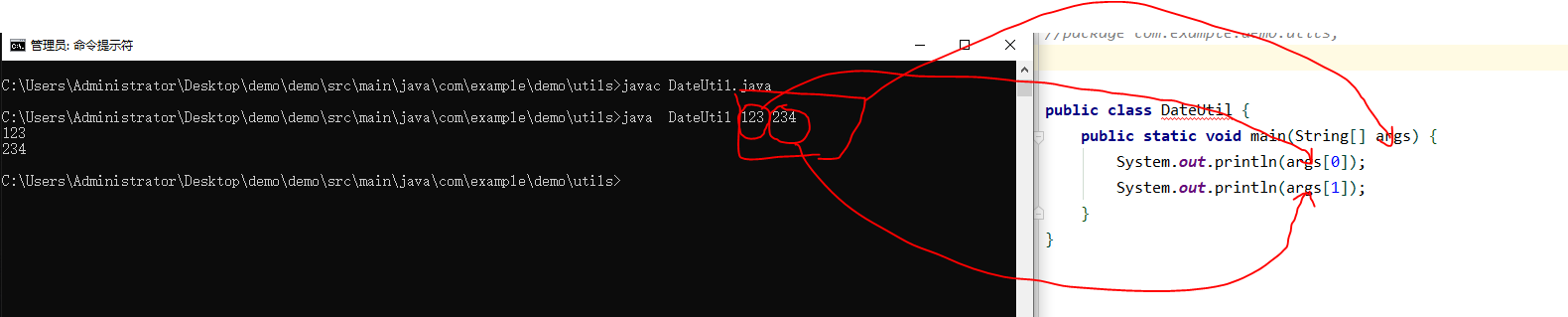


(2)java –Djava.ext.dirs=lib myClass

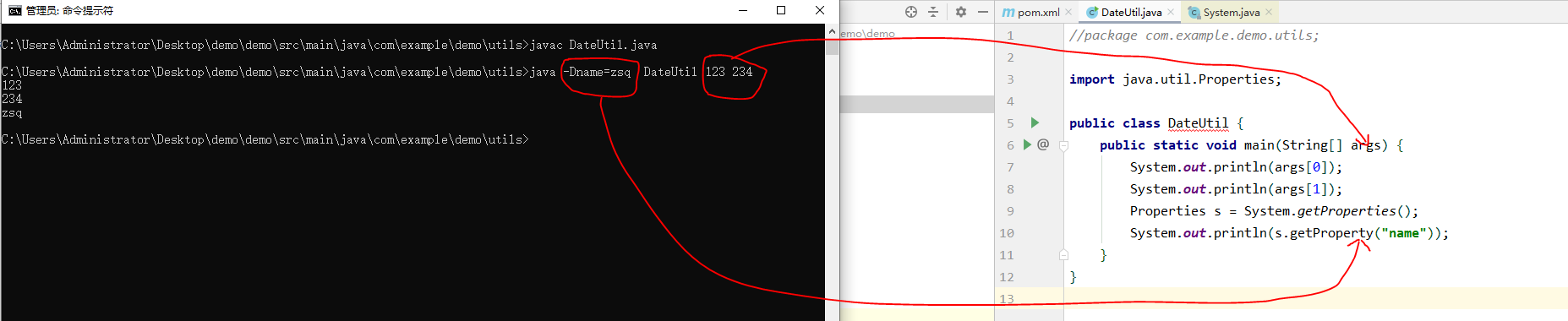
该用法可以简化加入多个jar包，java.ext.dirs是系统默认的系统属性名，表示外部依赖拓展包的目录。

2、传递主函数运行中的参数(args[])

该种传参方式是向主程序传入变量，常用于测试。



两种方式结合一起传参：



注意：系统参数在类名的前面，主函数中的参数在类名的后面。

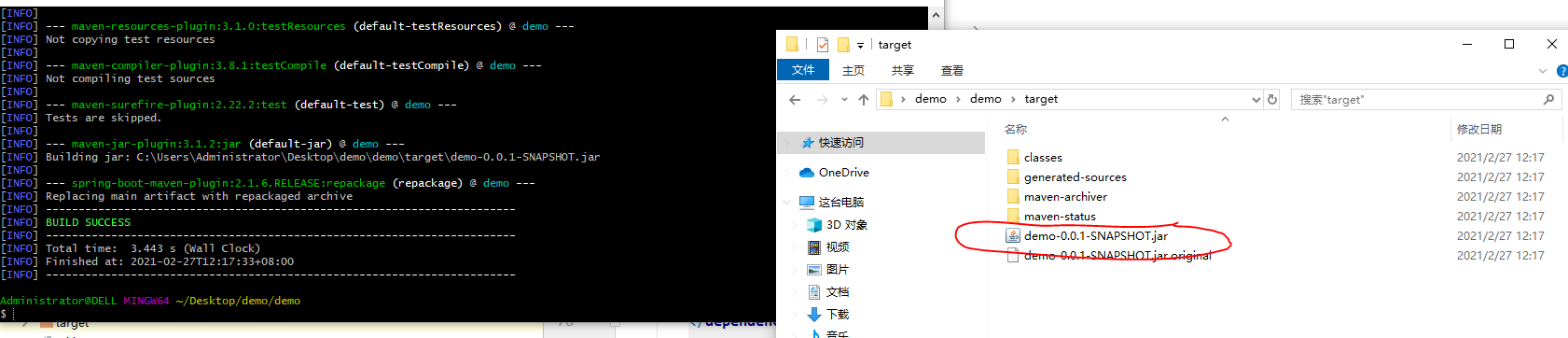
# #SpringBoot项目运行jar传参

1、引入依赖

***<*plugins*>  
 <*plugin*>  
 <*groupId*>***org.springframework.boot***</*groupId*>  
 <*artifactId*>***spring-boot-maven-plugin***</*artifactId*>  
 </*plugin*>  
</*plugins*>***

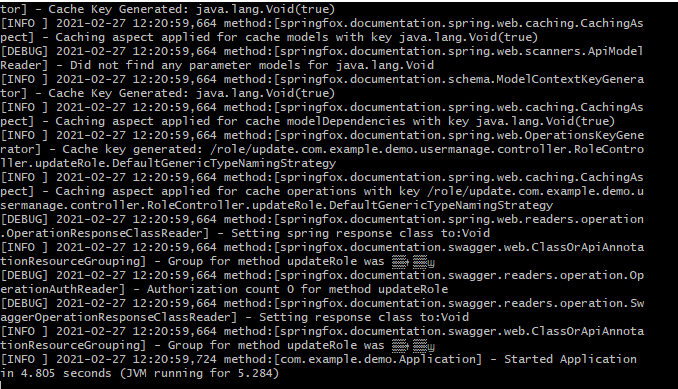
2、打包

mvn clean package -Dmaven.test.skip=true -T 2C



3、运行

java -jar demo-0.0.1-SNAPSHOT.jar



4、传参

java -jar demo-0.0.1-SNAPSHOT.jar --spring.profiles.active=dev

在命令行运行时，连续的两个减号--就是对application.properties中的属性值进行赋值的标识。故上述命令等价于在application.properties中添加属性spring.profiles.active=dev。

为了安全，Springboot提供了屏蔽命令行访问属性的设置，只需要设置：

SpringApplication.setAddCommandLineProperties(false)。

# #引入的其他jar包，命令行运行java程序

用java命令行直接执行程序，如果这个程序需要引用外部jar包；就不能单纯用java xx来执行。

解决方案：

1、javac/java –cp

Windows:

编译：javac -cp D:\yy\yy.jar;D\xx\xx.jar test.java

执行：java -cp D:\yy\yy.jar;D\xx\xx.jar test

Linux:

编译：javac -cp /yy/yy.jar:/xx/xx.jar test.java

执行：java -cp /yy/yy.jar:/xx/xx.jar test

原理：

java 是执行编译后的class的.

-cp 就是引入外部 zip包或者jar包，就相当于在环境变量配置的classpath后的参数。然后到指定的目录找指定运行时需要的class。

缺点：当依赖的包很多时，要一个个填写就很麻烦。接下第二种方案就可以解决这个问题。

2、javac/java -Djava.ext.dirs=

Windows:

编译：javac -Djava.ext.dirs=D:\usrlib;D:\lib test.java

执行：java -Djava.ext.dirs=D:\usrlib;D:\lib test

Linux:

编译：javac -Djava.ext.dirs=/usrlib:/lib test.java

执行：java -Djava.ext.dirs=/usrlib:/lib test

3、-cp 和 -Djava.ext.dirs区别

-cp 和 -classpath 一样，是指定java类运行所依赖其他类的路径，通常是类库，jar包之类，需要全路径到jar包，window上使用分号“;”  分隔，linux上是使用冒号“:”分隔。不支持通配符，也不支持文件夹的方式，需要列出所有jar包，用点“.”代表当前路径，这个如果要引用好多jar包，要一个个加入到路径里，写起来就很麻烦。

-Djava.ext.dirs是通过设置系统属性的方式也加载jar包的，级别就有点高，和-classpath的区别在于-Djava.ext.dirs会覆盖Java本身的ext设置，java.ext.dirs指定的目录由ExtClassLoader加载器加载，如果自定义的程序没有指定该系统属性（-Djava.ext.dirs=sss/lib）那么该加载器默认加载$JAVA\_HOME/lib/ext目录下的所有jar文件。

但如果手动指定系统属性且忘了把$JAVA\_HOME/lib/ext路径给加上，那么ExtClassLoader不会去加载$JAVA\_HOME/lib/ext下面的jar文件，意味着将失去一些功能。

例如java自带的加解密算法实现。具体说是jre的ext目录：D:\java\jdk1.8.0\_40\jre\lib\ext。一般情况下普通程序运行并无差异，因为可以看到ext目录下也就几个jar，但是如果使用java.security相关类时，也就是使用加密解决相关的算法，就会发现-Djava.ext.dirs会导致library找不到相关类而报错。如报错：

NoSuchAlgorithmException: Cannot find any provider supporting RSA

当然也有解决方法：

1、将ext下相关jar包复制到新的ext director。

2、在-D.java.ext.dirs中配置多个目录。可以使用冒号分隔（windows下使用分号）。比如：-Djava.ext.dirs=directoryA:directoryB