<https://blog.csdn.net/kyle_api/article/details/70137989>

如何把 java 程序编译成 .exe 文件。通常回答只有两种，一种是制作一个可执行的 JAR 文件包，然后就可以像.chm 文档一样双击运行了；而另一种是使用 JET 来进行 编译。但是 JET 是要用钱买的，而且据说 JET 也不是能把所有的 Java 程序都编译成执行文件，性能也要打些折扣。所以，使用制作可执行 JAR 文件包的方法就是最佳选择了，何况它还能保持 Java 的跨平台特性。   
  
下面就来看看什么是 JAR 文件包吧：   
  
1. JAR 文件包   
  
JAR 文件就是 Java Archive File，顾名思意，它的应用是与 Java 息息相关的，是 Java 的一种文档格式。JAR 文件非常类似 ZIP 文件——准确的说，它就是 ZIP 文件，所以叫它文件包。JAR 文件与 ZIP 文件唯一的区别就是在 JAR 文件的内容中，包含了一个 META-INF/MANIFEST.MF 文件，这个文件是在生成 JAR 文件的时候自动创建的。举个例子，如果我们具有如下目录结构的一些文件：   
  
==   
  
`-- test   
  
`-- Test.class   
  
把它压缩成 ZIP 文件 test.zip，则这个 ZIP 文件的内部目录结构为：   
  
test.zip   
  
`-- test   
  
`-- Test.class   
  
如果我们使用 JDK 的 jar 命令把它打成 JAR 文件包 test.jar，则这个 JAR 文件的内部目录结构为：   
  
test.jar   
  
|-- META-INF   
  
| `-- MANIFEST.MF   
  
`-- test   
  
`--Test.class   
  
2. 创建可执行的 JAR 文件包   
  
制作一个可执行的 JAR 文件包来发布你的程序是 JAR 文件包最典型的用法。   
  
Java 程序是由若干个 .class 文件组成的。这些 .class 文件必须根据它们所属的包不同而分级分目录存放；运行前需要把所有用到的包的根目录指定给 CLASSPATH 环境变量或者 java 命令的 -cp 参数；运行时还要到控制台下去使用 java 命令来运行，如果需要直接双击运行必须写 Windows 的批处理文件 (.bat) 或者 Linux 的 Shell 程序。因此，许多人说，Java 是一种方便开发者苦了用户的程序设计语言。   
  
其实不然，如果开发者能够制作一个可执行的 JAR 文件包交给用户，那么用户使用起来就方便了。在 Windows 下安装 JRE (Java Runtime Environment) 的时候，安装文件会将 .jar 文件映射给 javaw.exe 打开。那么，对于一个可执行的 JAR 文件包，用户只需要双击它就可以运行程序了，和阅读 .chm 文档一样方便 (.chm 文档默认是由 hh.exe 打开的)。那么，现在的关键，就是如何来创建这个可执行的 JAR 文件包。   
  
创建可执行的 JAR 文件包，需要使用带 cvfm 参数的 jar 命令，同样以上述 test 目录为例，命令如下：   
  
jar cvfm test.jar manifest.mf test   
  
这里 test.jar 和 manifest.mf 两个文件，分别是对应的参数 f 和 m，其重头戏在 manifest.mf。因为要创建可执行的 JAR 文件包，光靠指定一个 manifest.mf 文件是不够的，因为 MANIFEST 是 JAR 文件包的特征，可执行的 JAR 文件包和不可执行的 JAR 文件包都包含 MANIFEST。关键在于可执行 JAR 文件包的 MANIFEST，其内容包含了 Main-Class 一项。这在 MANIFEST 中书写格式如下：   
  
Main-Class: 可执行主类全名(包含包名)   
  
例如，假设上例中的 Test.class 是属于 test 包的，而且是可执行的类 (定义了 public static void main(String[]) 方法)，那么这个 manifest.mf 可以编辑如下：   
  
Main-Class: test.Test <回车>;   
  
这个 manifest.mf 可以放在任何位置，也可以是其它的文件名，只需要有 Main-Class: test.Test 一行，且该行以一个回车符结束即可。创建了 manifest.mf 文件之后，我们的目录结构变为：   
  
==   
  
|-- test   
  
| `-- Test.class   
  
`-- manifest.mf   
  
这时候，需要到 test 目录的上级目录中去使用 jar 命令来创建 JAR 文件包。也就是在目录树中使用“==”表示的那个目录中，使用如下命令：   
  
jar cvfm test.jar manifest.mf test   
  
之后在“==”目录中创建了 test.jar，这个 test.jar 就是执行的 JAR 文件包。运行时只需要使用 java -jar test.jar 命令即可。   
  
需要注意的是，创建的 JAR 文件包中需要包含完整的、与 Java 程序的包结构对应的目录结构，就像上例一样。而 Main-Class 指定的类，也必须是完整的、包含包路径的类名，如上例的 test.Test；而且在没有打成 JAR 文件包之前可以使用 java <类名>; 来运行这个类，即在上例中 java test.Test 是可以正确运行的 (当然要在 CLASSPATH 正确的情况下)。   
  
3. jar 命令详解   
  
jar 是随 JDK 安装的，在 JDK 安装目录下的 bin 目录中，Windows 下文件名为 jar.exe，Linux 下文件名为 jar。它的运行需要用到 JDK 安装目录下 lib 目录中的 tools.jar 文件。不过我们除了安装 JDK 什么也不需要做，因为 SUN 已经帮我们做好了。我们甚至不需要将 tools.jar 放到 CLASSPATH 中。   
  
使用不带任何的 jar 命令我们可以看到 jar 命令的用法如下：   
  
jar {ctxu}[vfm0M] [jar-文件] [manifest-文件] [-C 目录] 文件名 ...   
  
其中 {ctxu} 是 jar 命令的子命令，每次 jar 命令只能包含 ctxu 中的一个，它们分别表示：   
  
-c 创建新的 JAR 文件包   
  
-t 列出 JAR 文件包的内容列表   
  
-x 展开 JAR 文件包的指定文件或者所有文件   
  
-u 更新已存在的 JAR 文件包 (添加文件到 JAR 文件包中)   
  
[vfm0M] 中的选项可以任选，也可以不选，它们是 jar 命令的选项参数   
  
-v 生成详细报告并打印到标准输出   
  
-f 指定 JAR 文件名，通常这个参数是必须的   
  
-m 指定需要包含的 MANIFEST 清单文件   
  
-0 只存储，不压缩，这样产生的 JAR 文件包会比不用该参数产生的体积大，但速度更快   
  
-M 不产生所有项的清单（MANIFEST〕文件，此参数会忽略 -m 参数   
  
[jar-文件] 即需要生成、查看、更新或者解开的 JAR 文件包，它是 -f 参数的附属参数   
  
[manifest-文件] 即 MANIFEST 清单文件，它是 -m 参数的附属参数   
  
[-C 目录] 表示转到指定目录下去执行这个 jar 命令的操作。它相当于先使用 cd 命令转该目录下再执行不带 -C 参数的 jar 命令，它只能在创建和更新 JAR 文件包的时候可用。   
  
文件名 ... 指定一个文件/目录列表，这些文件/目录就是要添加到 JAR 文件包中的文件/目录。如果指定了目录，那么 jar 命令打包的时候会自动把该目录中的所有文件和子目录打入包中。   
  
下面举一些例子来说明 jar 命令的用法：   
  
1) jar cf test.jar test   
  
该命令没有执行过程的显示，执行结果是在当前目录生成了 test.jar 文件。如果当前目录已经存在 test.jar，那么该文件将被覆盖。   
  
2) jar cvf test.jar test   
  
该命令与上例中的结果相同，但是由于 v 参数的作用，显示出了打包过程，如下：   
  
标明清单(manifest)   
  
增加：test/(读入= 0) (写出= 0)(存储了 0%)   
  
增加：test/Test.class(读入= 7) (写出= 6)(压缩了 14%)   
  
3) jar cvfM test.jar test   
  
该命令与 2) 结果类似，但在生成的 test.jar 中没有包含 META-INF/MANIFEST 文件，打包过程的信息也略有差别：   
  
增加：test/(读入= 0) (写出= 0)(存储了 0%)   
  
增加：test/Test.class(读入= 7) (写出= 6)(压缩了 14%)   
  
4) jar cvfm test.jar manifest.mf test   
  
运行结果与 2) 相似，显示信息也相同，只是生成 JAR 包中的 META-INF/MANIFEST 内容不同，是包含了 manifest.mf 的内容   
  
5) jar tf test.jar   
  
在 test.jar 已经存在的情况下，可以查看 test.jar 中的内容，如对于 2) 和 3) 生成的 test.jar 分别应该此命令，结果如下；   
  
对于 2)   
  
META-INF/   
  
META-INF/MANIFEST.MF   
  
test/   
  
test/Test.class   
  
对于 3)   
  
test/   
  
test/Test.class   
  
6) jar tvf test.jar   
  
除显示 5) 中显示的内容外，还包括包内文件的详细信息，如：   
  
0 Wed Jun 19 15:39:06 GMT 2002 META-INF/   
  
86 Wed Jun 19 15:39:06 GMT 2002 META-INF/MANIFEST.MF   
  
0 Wed Jun 19 15:33:04 GMT 2002 test/   
  
7 Wed Jun 19 15:33:04 GMT 2002 test/Test.class   
  
7) jar xf test.jar   
  
解开 test.jar 到当前目录，不显示任何信息，对于 2) 生成的 test.jar，解开后的目录结构如下：   
  
==   
  
|-- META-INF   
  
| `-- MANIFEST   
  
`-- test   
  
`--Test.class   
  
8) jar xvf test.jar   
  
运行结果与 7) 相同，对于解压过程有详细信息显示，如：   
  
创建：META-INF/   
  
展开：META-INF/MANIFEST.MF   
  
创建：test/   
  
展开：test/Test.class   
  
9) jar uf test.jar manifest.mf   
  
在 test.jar 中添加了文件 manifest.mf，此使用 jar tf 来查看 test.jar 可以发现 test.jar 中比原来多了一个 manifest。这里顺便提一下，如果使用 -m 参数并指定 manifest.mf 文件，那么 manifest.mf 是作为清单文件 MANIFEST 来使用的，它的内容会被添加到 MANIFEST 中；但是，如果作为一般文件添加到 JAR 文件包中，它跟一般文件无异。   
  
10) jar uvf test.jar manifest.mf   
  
与 9) 结果相同，同时有详细信息显示，如：   
  
增加：manifest.mf(读入= 17) (写出= 19)(压缩了 -11%)   
  
4. 关于 JAR 文件包的一些技巧   
  
1) 使用 unzip 来解压 JAR 文件   
  
在介绍 JAR 文件的时候就已经说过了，JAR 文件实际上就是 ZIP 文件，所以可以使用常见的一些解压 ZIP 文件的工具来解压 JAR 文件，如 Windows 下的 WinZip、WinRAR 等和 Linux 下的 unzip 等。使用 WinZip 和 WinRAR 等来解压是因为它们解压比较直观，方便。而使用 unzip，则是因为它解压时可以使用 -d 参数指定目标目录。   
  
在解压一个 JAR 文件的时候是不能使用 jar 的 -C 参数来指定解压的目标的，因为 -C 参数只在创建或者更新包的时候可用。那么需要将文件解压到某个指定目录下的时候就需要先将这具 JAR 文件拷贝到目标目录下，再进行解压，比较麻烦。如果使用 unzip，就不需要这么麻烦了，只需要指定一个 -d 参数即可。如：   
  
unzip test.jar -d dest/   
  
2) 使用 WinZip 或者 WinRAR 等工具创建 JAR 文件   
  
上面提到 JAR 文件就是包含了 META-INF/MANIFEST 的 ZIP 文件，所以，只需要使用 WinZip、WinRAR 等工具创建所需要 ZIP 压缩包，再往这个 ZIP 压缩包中添加一个包含 MANIFEST 文件的 META-INF 目录即可。对于使用 jar 命令的 -m 参数指定清单文件的情况，只需要将这个 MANIFEST 按需要修改即可。   
  
3) 使用 jar 命令创建 ZIP 文件   
  
有些 Linux 下提供了 unzip 命令，但没有 zip 命令，所以需要可以对 ZIP 文件进行解压，即不能创建 ZIP 文件。如要创建一个 ZIP 文件，使用带 -M 参数的 jar 命令即可，因为 -M 参数表示制作 JAR 包的时候不添加 MANIFEST 清单，那么只需要在指定目标 JAR 文件的地方将 .jar 扩展名改为 .zip 扩展名，创建的就是一个不折不扣的 ZIP 文件了，如将上一节的第 3) 个例子略作改动：   
  
jar cvfM test.zip test

1.在dos下面打包jar文件   
  
D:\jdk1.5.0\_07\bin> jar  /help   
用法：jar  {ctxu}[vfm0Mi]  [jar-文件]  [manifest-文件]  [-C  目录]  文件名  ...   
选项：   
-c  创建新的存档   
-t  列出存档内容的列表   
-x  展开存档中的命名的（或所有的〕文件   
-u  更新已存在的存档   
-v  生成详细输出到标准输出上   
-f  指定存档文件名   
-m  包含来自标明文件的标明信息   
-0  只存储方式；未用ZIP压缩格式   
-M  不产生所有项的清单（manifest〕文件   
-i  为指定的jar文件产生索引信息   
-C  改变到指定的目录，   
在命令行中运行刚刚生成的jar，命令如下：   
java -jar test.jar   
如果在jar中有一些System.out.prinln语句，运行后想看看打印出的内容，可以用一下命令：   
java -jar test.jar >log.txt   
输出信息会被打印到log.txt中，当然log.txt自动生成，并位于和test.jar一个目录中。   
这里也可以新建一个start.bat文件，创建后；右键---编辑，输入:java -jar test.jar >log.txt；以后直接双击start.bat即可运行test.jar啦。   
注意，如果你是在文件导出放在如：E盘，则你需要导入得包也要放在这个目录里。   
如果一个文件名是一个目录，它将被递归处理。   
清单（manifest〕文件名和存档文件名都需要被指定，按 'm '  和  'f '标志指定的相同顺序。

示例1：将两个class文件存档到一个名为  'classes.jar '  的存档文件中：   
jar  cvf  classes.jar  Foo.class  Bar.class   
示例2：用一个存在的清单（manifest）文件  'mymanifest '  将  foo/  目录下的所有   
文件存档到一个名为  'classes.jar '  的存档文件中：   
jar  cvfm  classes.jar  mymanifest  -C  foo/  .

2.用eclipse打包jar文件

Eclipse通过导出的方式(右键单击项目，之后选择Export)打包java类文件生成jar包。

方法一：(在项目工程没有引用外部jar包时，直接导出)   
选中工程---->右键，Export...--->Java--->JAR file--->next-->选择jar file的路径及名称-->next-->next---> 选择main class--->finish.   
方法2：   
当工程引用了其他的外部jar时，由于eclipse不支持同时导出外部jar包的功能，所以比较麻烦一点；具体步骤如下：   
生成manifest file:选中工程--->右键，Export--->Java--->JAR file--->选择jar file的路径及名称--->next--->next--->next；

Android如何将程序打成jar包

导出  
1. 创建java project  
2. 工程属性java build path->Libraries中加入android.jar  
3. package explorer中右键点击工程，export java->jar file.  
  
使用  
1. 创建android project  
2. 工程属性java build path->Libraries中加入刚才的jar文件  
3. 在android工程中使用jar的类。  
  
只能在jar中共享代码，使用资源不行。  
  
--------------------------------------------------------------------------------  
  
如果将自己的代码封装成一个jar库供其他程序复用呢?   
首先创建一个Android工程，确定可以编译通过，  
移除androidmanifest.xml文件，和相关的程序资源图片。  
在Eclipse中我们找到Package Explorer，选择export，  
接下来在导出对话框中选择Java，JAR file，  
在JAR File Specification这里选择我们需要到处的资源等等，  
在JAR file:路径选择我们导出后文件的存放位置，  
在下面的Compress the contents of the JAR file勾选可以压缩生成文件的体积。  
最后单击Next或Finish即可完成导出，  
这样可以在以后的工程中导入这个库，起到了代码复用的效果类似 Windows 32开发的Dll文件一样。