



黑客派

首发! JDK14 之 jpackage 命令尝鲜

作者: [xiaodaojava](#)

原文链接: <https://hacpai.com/article/1583033951839>

来源网站: 黑客派

许可协议: [署名-相同方式共享 4.0 国际 \(CC BY-SA 4.0\)](#)



jpackage命令使用场景

使用场景是面向java桌面端程序打包. 可以让windows/mac 直接双击使用java程序,对系统里面有没有dk/jre不做要求.

在这里,我们先理一下以前java桌面程序是怎么运行的:

jdk8及以前

这时候的java客户端程序,主要分两部分,一部分是可执行jar包,一部分是电脑里面要装jdk/jre , 然后通过

`java -jar XXXXX.jar` 来运行程序.

jdk9及以后的模块化

这是java模块化的一个分界线,其实jdk9,jdk10,我都没有用过, 我是直接尝试的jdk11. 在模块化的基础, 项目多了一个文件, `module-info.java` 如下图所示:

```
module red.lixiang.tools.desktopmain {  
    requires fastjson;  
    requires javafx.controls;  
    requires javafx.fxml;  
    requires red.lixiang.tools.common;  
    requires red.lixiang.tools.base;  
    requires com.jfoenix;  
  
    opens com.platform.tools.desktop to javafx.fxml;  
    opens com.platform.tools.desktop.controller.kub to javafx.fxml;  
    opens com.platform.tools.desktop.controller.host to javafx.fxml;  
    opens com.platform.tools.desktop.controller.set to javafx.fxml;
```

```
opens com.platform.tools.desktop.conf to fastjson;

exports com.platform.tools.desktop;
exports com.platform.tools.desktop.controller;
exports com.platform.tools.desktop.conf;

}
```

如上图,所示,主要是 requires , opens , exports 这三个关键字.

requires : 当前模块想要运行的话,需要哪些依赖,相当于在maven引包的基础上,要再声明一遍依赖

opens: 当前模块中的哪些包,可以被反射. 如fxml中要声明处理的controller , 每当fxml加载的时候,都去新建一个controller实例, 所以需要opens controller的包名给javafx.fxml

exports: 当前模块有哪些包,可以被引用. 一般用于别的模块依赖当前模块的时候

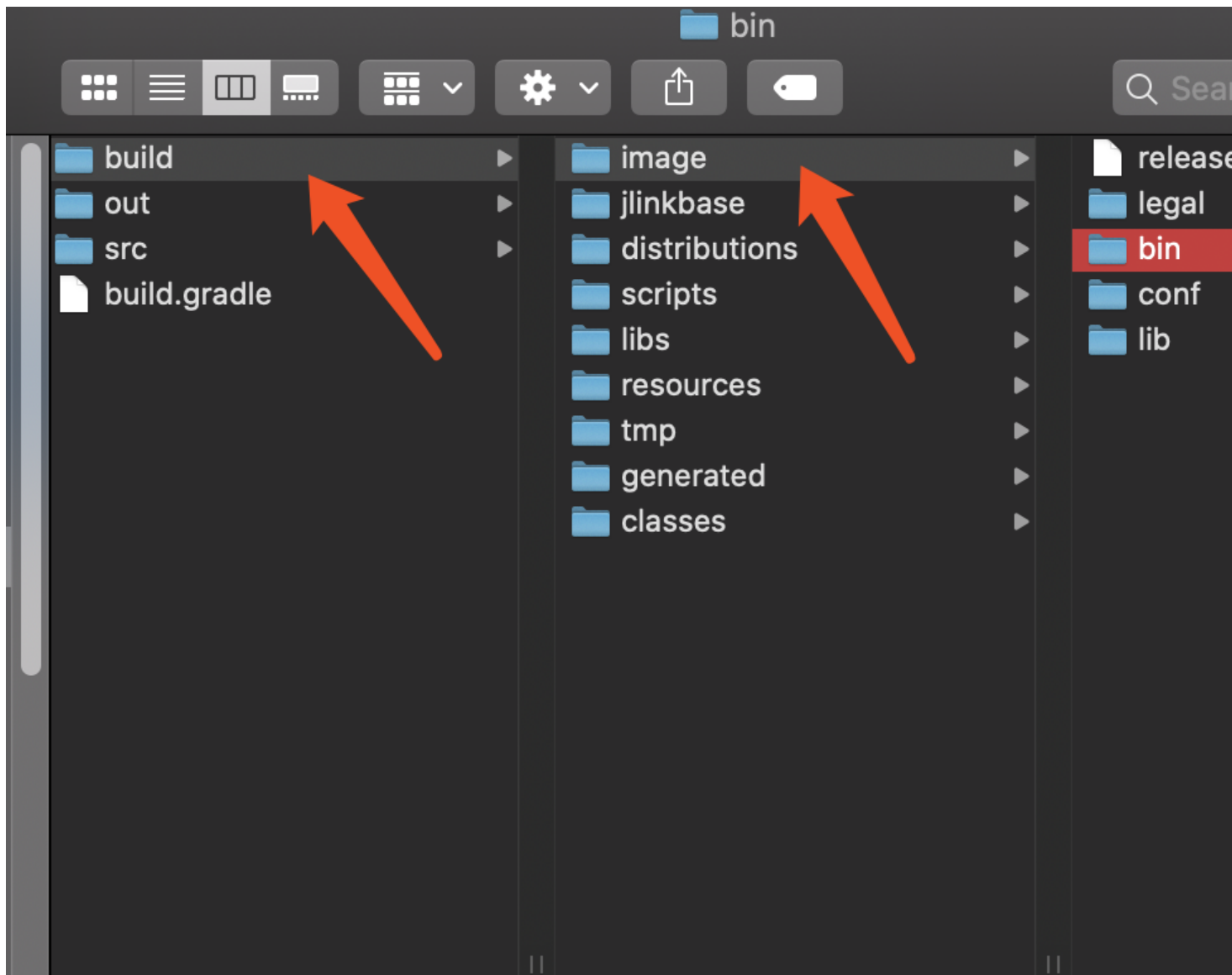
jdk9及以后的打包方式

在jdk9及以后,打包方式,可以不局限于打jar包,可以把jdk的这些模块给一起用jlink打成一个镜像.

如下我用gradle的jlink去打包

- ▼  tools-desktop
 - ▶  Source Sets
 - ▼  Tasks
 - ▼  application
 -  run
 - ▼  build
 -  assemble
 -  build
 -  buildDependents
 -  buildNeeded
 -  classes
 -  clean
 -  createDelegatingModules
 -  createMergedModule
 -  jar
 -  **jlink**
 -  jlinkZip
 -  jpackage
 -  jpackageImage
 -  prepareMergedJarsDir
 -  prepareModulesDir
 -  suggestMergedModuleInfo

最后的成果:



如上所示, 会在build目录,生面image镜像文件夹,然后在其中有bin , conf, lib .然后用cmd运行bin目录中的Sunflower就可以了, 如下所示: 记住这朵Sunflower , 我要让他盛开在广大java程序员的桌面,哈哈

```
./Sunflower (java)

# lixiang @ localhost in ~ - ssh - shenzhen/tools/tools-de
in on git:jdk11 x [10:47:29]
$ pwd
/Users/lixiang /tools/tools-desktop/build/

# lixiang @ localhost in ~ - ssh - shenzhen/tools/tools-de
in on git:jdk11 x [10:47:31]
$ ls
Sunflower      java      javadoc      jrunscript   serial
Sunflower.bat  javac     jdb          keytool

# lixiang @ localhost in ~ - ssh - shenzhen/tools/tools-de
in on git:jdk11 x [10:47:33]
$ ./Sunflower
█
```

可以看到, 这时候除了程序主体存在, 还有一个cmd窗口在这里, 何其不方便!!! 这时候, jdk13及以后的jpackage命令就派上用场了, 刚好我写这个的时候, jdk14 early access 版本发布了, 就直接拿jdk14做了实验

JDK14的jpackage命令

在安装好jdk14之后, 可以使用jpackage -h 命令看下使用说明, 如下图所示

```
# lixiang @ localhost in ~
in on git:jdk11 x [10:51:02] C: 150
$ /Users/lixiang/soft/jdk-14.jdk/Contents/Home/bin/jpackage
WARNING: Using incubator modules: jdk.incubator.jpackage
Usage: jpackage <options>

Sample usages:
-----

Generate an application package suitable for the host system:
  For a modular application:
    jpackage -n name -p modulePath -m moduleName/className
  For a non-modular application:
    jpackage -i inputDir -n name \
      --main-class className --main-jar myJar.jar
  From a pre-built application image:
    jpackage -n name --app-image appImageDir
Generate an application image:
  For a modular application:
    jpackage --type app-image -n name -p modulePath \
      -m moduleName/className
  For a non-modular application:
    jpackage --type app-image -i inputDir -n name \
      --main-class className --main-jar myJar.jar
```

其实这个是在上一部Jlink有镜像之后,对镜像做了一层封装.

我最后实验成功的打包命令是:

```
/Users/lixiang/soft/jdk-14.jdk/Contents/Home/bin/jpackage --runtime-image image --type
mg --name Sunflower --module red.lixiang.tools.desktopmain/com.platform.tools.desktop.D
sktopMain
```

--runtime-image 后面跟的是文件夹, 就是我们build目录下的image文件夹

--type 打成什么样的包, 在mac系统中有 app-image , dmg , pkg三个选项, windows的我没试, 应该有exe选项

--name 软件的名字, 这里是Sunflower(向日葵)

--moudle 模块的名字, 相当于以前的Main函数的格式, 这里换成了模块 '包名'/Main函数名

运行后结果如下:

```
..desktop/build (zsh)

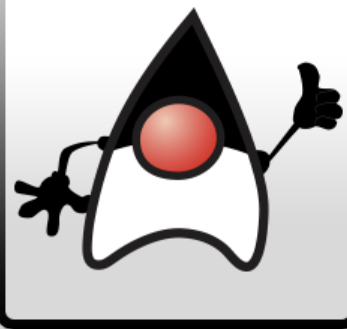
# lixiang @ localhost in ~ tools-de
jdk11 x [11:19:33]
$ pwd
/Users/lixiang/360jk/code/shenzhen/tools/tools-desktop/build

# lixiang @ localhost in tools-de
jdk11 x [11:19:34]
$ ls
classes          generated        jlinkbase       resources        tmp
distributions    image          libs            scripts

# lixiang @ localhost in tools-de
jdk11 x [11:19:35]
$ /Users/lixiang/soft/jdk-14.jdk/Contents/Home/bin/jpackage
ge --type dmg --name Sunflower --module red.lixiang.tools.c
form.tools.desktop.DesktopMain
WARNING: Using incubator modules: jdk.incubator.jpackage

# lixiang @ localhost in tools-de
jdk11 x [11:19:54]
$
```

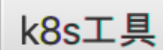
最终如上图,可以看到生成的dmg文件, 在mac上,进行安装就可以使用了



Sunflower



Applications



HOST工具

设置

新增配置

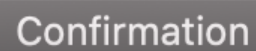
刷新

☒ 测试

e

r

```
# welcome to sunflower!  
127.0.0.1 a.b.c
```



Confirmation

电脑管理员密码:

Cancel

OK

原文链接: [首发! JDK14 之 jpackage 命令尝鲜](#)

项目源码

一.Sunflower应用还在开发中, 不久后会开放下载.

二. javafx+gradle应用: <https://openjfx.io/openjfx-docs/#gradle>

三. 项目所用到的判断系统工具类,读取/更改host工具类等开源地址如下:

<https://github.com/xiaodaojava/sunshine>

里面还有很多很多常用的工具类. 求一个star~~~