gradleMakeWindowsBat

Skeleton gradle project template to build an executable java jar plus all distributable assets for a 'HelloWorld' java source. This is to deploy your app as a stand-alone app on a client system, something like a GUI, or batch job app - **NOT** a web service.

The tasks to do this are a freebie when a *build.gradle* file includes a plugin module named 'application' courtesy of

```
apply plugin:'application'
```

Gradle Build Tool

This project includes a full gradle build tool as a 'wrapper' bit inside this download. So there's no need install gradle on your system. You can run this app once you've done a git clone.

Git Source Code Control

Yes, you need **Git** installed on your system to get started. So change into a new folder/directory on your local system and do this:

git clone https://github.com/jnorthr/gradleMakeWindowsBat.git

This makes a project directory folder named **gradleMakeWindowsBat** so **cd** into that.

You could use this approach as a foundation to create several brand-new projects. Sure you would need to change **gradleMakeWindowsBat** to something else and the main class names would change too, but a lot of the setup work is done for you here.

Build / Check

Now you can just run the gradle wrapper like so:

```
gradlew check
```

or

```
bash ./gradlew check
```

This will ask gradle to download any dependency bits it needs to make it happy,

and check that everything is cool on your installed version.

Build

Ok, to make it all happen, run gradlew again without options.

```
gradlew
```

or

```
bash ./gradlew
```

The default tasks are run. These are:

```
defaultTasks 'clean','build','javadoc','installApp', 'startScripts', 'fatJar',
'distTar', 'distZip', 'run'
```

Tasks to Deploy An App

- *clean* - gradle cleans the environment before starting
- *build* - when javac - compiles the HelloWorld module
- *javadoc* - the javadoc API document is created with any source code comments
- *installApp* makes a folder named **install** and a sub-folder of **gradleMakeWindowsBat** containing /bin, /docs, /lib folders with all the pieces to do a full install on a client system. Ship the **gradleMakeWindowsBat** folder and change the client's OS path variable to include **gradleMakeWindowsBat/bin** then on the command line you/they can run this job by typing the name of this batch script file as **gradleMakeWindowsBat** - *easy-peasy* !
- *startScripts* - makes a folder of identical windoze and unix batch script files. These scripts will run the core job noted in the **build.gradle** file as *mainClassName = "com.jnorthr.DateUtils"* - yeah, you can change that for your own needs. Be sure to add the .jar files in **build/libs** to the system CLASSPATH variable.
- *fatJar* - my favorite ! makes an executable jar file of the java/groovy/scala/jvm language classes and bits. You can just double click on the **gradleMakeWindowsBat-all-1.0.jar** file to run your app.

  Alternatively from a command line with the jar in the same folder, you can

do this:

```
java -jar gradleMakeWindowsBat-all-1.0.jar
```

- Everything needed has been included in this 'fat' jar !

Tasks to Transfer The Project Build Environment

When we need to move/copy/transfer a complete project that includes all the development tools, build stuff and source code controls, we can use either of these two tasks depending on the target computer system's OS:

- *distTar* packages up the full folder directory and makes it ready for shipment to a target computer system running Unix/Linux/Ubuntu etc.
- *distZip* packages up the full project folder directory in a compressed archive format. This is useful as a general-purpose tool to move a complete project around to target computer systems where the OS is unconventional like, say, Windows, IBM, DEC.

Both tasks run as a default set of tasks. Project build times may seem excessive as so many solutions are being produced. To speed things up you could alter the **build.gradle** file to remove these two tasks if you do not plan to move the project to another system.

Task to Run This App

- the *run* task starts execution of an app. The app name is declared in the **build.gradle** file as

```
mainClassName = "com.jnorthr.DateUtils"
```

change to suit your own requirements.