Deliver Your Java Application in One-JAR™!

<<Background Home Print Layout Ant Taskdef>>

Quick Start

There are various approaches to getting started with One-JAR. Ant users will find the "Application Generator Approach" most useful, Maven users the "Maven Approach", command-line tool users may prefer the "Command Line Approach".

Application Generator Approach

This approach provides you with a complete Eclipse/Ant application directory, which you can use as a starting point for your own One-JAR application. The application generator is driven by a template built into the one-jar-appgen.jar file (see one-jar-appgen)

1. Download one-jar-appgen-0.97.jar
2. Generate application, build, and run it.

```
$ java -jar one-jar-appgen-0.97.jar

   Enter project path (project name is last segment): c:/tmp/test-one-jar
   Enter java package name: com.example.onejar

$ cd c:/tmp/test-one-jar
$ ant
$ cd build
$ java -jar test-one-jar.jar

   test_one_jar main entry point, args=[]
   test_one_jar main is running
   test_one_jar OK.
```

Add source code to the `src` directory, library jars to the `lib` directory, and rebuild.

Command-Line Approach

The use of Ant is not required: a One-JAR archive is simple to build using just the `jar` tool using the following steps.

1. Create a working directory to act as the "root" of the one-jar with `main`, `lib` sub-directories.
2. Copy your main application jar file into `root/main` and library dependencies into `root/lib`
3. Unjar the `one-jar-boot-0.97.jar` file into the root directory, and delete the "src" tree
4. Edit the boot-manifest.mf file and add a new line: `One-Jar-Main-Class: your-main-class`
5. cd root; jar -cvfm ../one-jar.jar boot-manifest.mf .

You should end up with a One-JAR archive which mirrors the "root" tree:

```
one-jar.jar
|  META-INF/MANIFEST.MF
|  .version
|  com/simontuffs/onejar
   |  Boot.class, ...etc.
|  doc/one-jar-license.txt
|  main/main.jar
|  lib/a.jar ...etc.
```

Thats it: no code to write, just a directory tree, some copy operations, and a file edit. The One-JAR classloader discovers the libraries and main code based on their position in the archive, and ignores any other Jar files should you need to embed archives which should not be on the classpath. Embedding the `one-jar-license.txt` ensures compliance with the BSD-style license.

Maven Approach

There is a Maven2 plugin for One-JAR. It is easy to use for Maven projects. Please consult the documentation here: http://code.google.com/p/onejar-maven-plugin/

Ant Taskdef Approach

Detailed use of the One-JAR Ant Taskdef is discussed here. Note that the `one-jar-appgen` approach uses the Ant taskdef.

SDK Approach

Use of the (deprecated) SDK is discussed here

If you like *One-JAR* then you might want to check out some of the other Open-Source projects developed by simontuffs.com: