



JAR Files and JAVA Extensions

How To Make
Your JAVA Software
Feel Professional

The perception of (dis)order...

- When we create a JAVA application, it is typically implemented as a set of JAVA objects with some global POE
 - Each object has it's own JAVA file...
 - ... which compiles into it's own CLASS file
- Most users are used to applications that are a single file or button click away



Enter the JAR file

- JAR files are archive files, like ZIP files
 - They can contain files and folders
 - They are automatically “compressed”
- There are also some additional features
 - JAR files can be digitally signed to verify the origin of the code in the file
 - JAR files can be “executed” as if they were a full fledged application

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited



The First Step...

- First thing you have to do is make sure that you have got your Package right
- Remember to invert your domain name to make globally unique package names
- package edu.columbia.cs.cgiui.mars
- This will lead to a mess of directories that... luckily RAD tools like Forte deals with this auto-magically

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited

Creating a JAR

- The jar tool works just like UNIX tar
 - `jar cvf MyApplication.jar file1 file2 ...`
- Typically you would change directory to the root of your development
 - `jar cvf MyApplication.jar *`
- The `-C` option allows more flexibility
 - `jar cvf MyApplication.jar -C /devel/root *`
- Be careful of source code inclusion!

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited

```
simon@sltp20: /tmp/jartest
[simon@sltp20 jartest]$ ls -la edu/columbia/cs3101-2/InstantMessenger/
total 34
drwxrwxr-x 2 simon simon 1024 Dec 9 01:42 .
drwxrwxr-x 3 simon simon 1024 Dec 9 01:39 ..
-rwxr-xr-x 1 simon simon 236 Dec 9 01:38 Callback.class
-rwxr-xr-x 1 simon simon 1603 Dec 9 01:38 CallbackImpl.class
-rwxr-xr-x 1 simon simon 1655 Dec 9 01:38 CallbackImpl_Skel.class
-rwxr-xr-x 1 simon simon 3006 Dec 9 01:38 CallbackImpl_Stub.class
-rwxr-xr-x 1 simon simon 572 Dec 9 01:38 Client1.class
-rwxr-xr-x 1 simon simon 599 Dec 9 01:38 Client2.class
-rwxr-xr-x 1 simon simon 599 Dec 9 01:38 Client3.class
-rwxr-xr-x 1 simon simon 4752 Dec 9 01:38 Client.class
-rwxr-xr-x 1 simon simon 614 Dec 9 01:38 ConnectDialog1.class
-rwxr-xr-x 1 simon simon 641 Dec 9 01:38 ConnectDialog2.class
-rwxr-xr-x 1 simon simon 3414 Dec 9 01:38 ConnectDialog.class
-rwxr-xr-x 1 simon simon 300 Dec 9 01:38 Server.class
-rwxr-xr-x 1 simon simon 2505 Dec 9 01:38 ServerImpl.class
-rwxr-xr-x 1 simon simon 1974 Dec 9 01:38 ServerImpl_Skel.class
-rwxr-xr-x 1 simon simon 3506 Dec 9 01:38 ServerImpl_Stub.class
[simon@sltp20 jartest]$ jar cvf InstantMessenger.jar *
added manifest
adding: edu/(in = 0) (out= 0)(stored 0%)
adding: edu/c/(in = 0) (out= 0)(stored 0%)
adding: edu/c/columbia/(in = 0) (out= 0)(stored 0%)
adding: edu/c/columbia/cs3101-2/(in = 0) (out= 0)(stored 0%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/(in = 0) (out= 0)(stored 0%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/Callback.class(in = 236) (out= 179)(deflated 24%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/CallbackImpl.class(in = 1603) (out= 880)(deflated 45%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/CallbackImpl_Skel.class(in = 1655) (out= 933)(deflated 43%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/CallbackImpl_Stub.class(in = 3006) (out= 1460)(deflated 51%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/Client1.class(in = 572) (out= 338)(deflated 40%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/Client2.class(in = 599) (out= 354)(deflated 40%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/Client3.class(in = 599) (out= 353)(deflated 41%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/Client.class(in = 4752) (out= 2341)(deflated 50%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/ConnectDialog1.class(in = 614) (out= 344)(deflated 43%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/ConnectDialog2.class(in = 641) (out= 358)(deflated 44%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/ConnectDialog.class(in = 3414) (out= 1699)(deflated 50%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/Server.class(in = 300) (out= 212)(deflated 29%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/ServerImpl.class(in = 2505) (out= 1385)(deflated 44%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/ServerImpl_Skel.class(in = 1974) (out= 1049)(deflated 46%)
adding: edu/c/columbia/cs3101-2/InstantMessenger/ServerImpl_Stub.class(in = 3506) (out= 1590)(deflated 54%)
[simon@sltp20 jartest]$
```

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited



Viewing/Extracting a JAR

- JAR file contents can be viewed by:
 - `jar tvf MyApplication.jar`
- The “t” stands for Table of Contents
- A JAR can also be extracted by using the “x” command
 - `jar xvf MyApplication.jar`
 - This automatically overwrites existing files!

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited



The Manifest

- If you create a JAR and view it, you will see a file called MANIFEST.MF in the table of contents
- This file is automatically created by the JAR tool and contains special information for the JAR
- To alter the manifest:
 - `jar cmvf manifest-addition MyApp.jar *`

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited



Manifest Manipulations

- Set the class with the point of entry (main) for this application:
 - Main-Class: PrimaryPointOfEntry
 - You can execute a JAR file by using the command “java -jar MyApplication.jar”
 - The Win32 JRE allows “double clicking”
- Import other JAR files:
 - Class-Path: place/OtherStuff.jar

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited



Version Control

- Name: ThePackage/MyApplication/
Sealed: true
Implementation-Title: “My Stuff”
Implementation-Version: “build57”
Implementation-Vendor: “My Company”
- Requires all classes of this application to be present in this JAR and saves some descriptive information with it

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited

Signed JARS

- Digital signatures verify the integrity and source of the JAR
- Use the “keytool” to manipulate your public and private keys
- Use “jarsigner” to actually sign the jar
 - jarsigner will overwrite your existing JAR file and replace it with the signed version

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited

Remote JARs

- You can run remote JARs with the JarRunner class...
 - `java JarRunner http://server/MyJar.jar`
- The `java.util.jar` package has utilities to download and use JAR files into your current VM
 - `JarClassLoader(http://server/MyJar.jar);`
 - `invokeClass("TheClassName", args);`

Copyright 1999-2002 Simon Lok Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited



JAVA Extensions

- The JRE can be extended using code that you write in JAVA...
 - A number of core technologies started out as extensions to the JRE
 - Commonly used extensions from Sun include JAVAMail (to access email via IMAP), JAVA3D (for 3D graphics) and JAXP (XML parsing and generation)

Copyright 1999-2002 Simon Lok. Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited.



Your Own Extensions

- Create a JAR that contains your code with the proper package statements setup for a global namespace
- Put the JAR in JRE_HOME/lib/ext
 - Win32 - c:\jdk1.3\jre\lib\ext\
 - Linux - /opt/jdk1.3/jre/lib/ext/
 - Solaris - /usr/java/jre/lib/ext/

Copyright 1999-2002 Simon Lok. Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited.



Good Practices

- Globalize your namespace!
- Separate your executable into libraries that can be reused and code for an application (front end)
 - Install your reusable code in jre/lib/ext
 - Put your front-end code into a JAR with a Main-Class attribute in the manifest
 - Sign and seal all of your JARs

Copyright 1999-2002 Simon Lok. Reproduction and/or redistribution in whole or part without written authorization is expressly prohibited.