



Java Básico

Empaquetamiento y despliegue de aplicaciones

Copyright

- Copyright (c) 2004
José M. Ordax
- Este documento puede ser distribuido solo bajo los términos y condiciones de la Licencia de Documentación de javaHispano v1.0 o posterior.
- La última versión se encuentra en
<http://www.javahispano.org/licencias/>

Empaquetar y desplegar

- Ya tenemos nuestra aplicación compilada y funcionando.
- ¿Cómo se la damos a los usuarios?
- ¿Qué le damos exactamente a los usuarios?
- ¿Y si no conocemos a los usuarios?
- Luego necesitamos una forma de empaquetar los aplicativos Java y de desplegarlos a las plataformas cliente.

Empaquetado

- La primera idea es separar el código fuente de los bytecode.
- El código fuente no es útil para el usuario final, y además, ocupa espacio aumentando el tamaño del aplicativo.
- Existen dos técnicas para realizar esta separación:
 - Manual cada vez que queramos empaquetar (desaconsejada).
 - Una buena estructura de directorios y la opción `-d` del compilador.

Estructura de directorios

- Existen múltiples posibilidades a la hora de organizar los directorios. Esta es una sugerencia:
 - Un primer directorio raíz.
Por ejemplo `c:\trabajo`
 - Un directorio para el código fuente.
Por ejemplo `c:\trabajo\src`
 - Un directorio para los bytecode.
Por ejemplo `c:\trabajo\bin`
- En el directorio `src` iremos creando todos los paquetes y código fuente (*.java).

La opción `-d` de `javac`

- Si compilamos tal cual el código, los bytecode se crearían en la estructura del código fuente.
- Para evitar esto, existe la opción `-d` con la que indicaremos el directorio donde queremos que nos genere el bytecode (*.class).

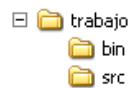
```
javac -d ..\bin *.java
```
- Es muy importante que el directorio de los fuentes (`c:\trabajo\src`) esté en el `CLASSPATH` para evitar posibles problemas.

La opción -d de javac

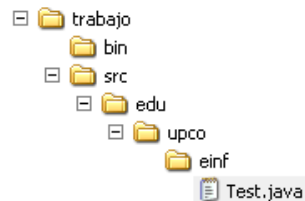
- Al analizar el contenido del directorio especificado en la opción -d veremos que no solo crea allí los bytecode, sino que también genera la estructura de directorios correspondiente a los paquetes Java.
- El procedimiento de ejecución de la aplicación no se ve modificado por esta forma de compilación.
- Simplemente habrá que seguir teniendo en cuenta que:
 - Si hay paquetes, el nombre de la clase incluye el paquete.
 - Añadir al CLASSPATH el directorio de los bytecode para evitar problemas (c:\trabajo\bin).

Ejemplo

- Estructura inicial:



- Desarrollo de los fuentes:



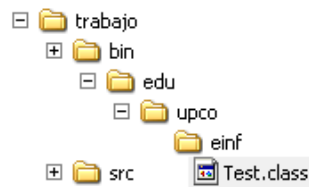
Ejemplo

1 Ajustamos el CLASSPATH:

2 `set CLASSPATH=c:\trabajo\src;c:\trabajo\bin;%CLASSPATH%`

3 Compilamos:

4 `cd c:\trabajo\src`
`javac -d ../bin edu\upco\einf\Test.java`



Ejemplo

1 Ejecutamos:

2 `java edu.upco.einf.Test`

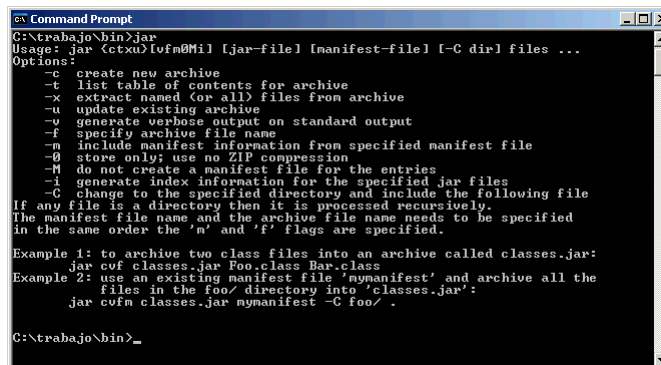
3 Nota: recuerda que ambos directorios: src y bin fueron añadidos al CLASSPATH.

Empaquetado

- Ya tenemos los fuentes separados de los bytecode.
- Ahora debemos empaquetar todos los bytecode para que sea más fácil el despliegue.
- Para ello vamos a usar los ficheros JAR:
 - JAR (Java Archive).
- Los ficheros JAR están basados en el formato de los ficheros ZIP y nos permiten empaquetar todas las clases en un único fichero.

Ficheros JAR

- Para trabajar con ficheros JAR, el SDK contiene una herramienta: jar.exe
- Si la ejecutamos tal cual, nos muestra todas sus opciones:



```
Command Prompt
C:\trabajo\bin>jar
Usage: jar <ctxu>[vfm0Ml] [-jar-file] [-manifest-file] [-C dir] files ...
Options:
  -c      create new archive
  -t      list table of contents for archive
  -x      extract named <or all> files from archive
  -u      update existing archive
  -v      generate verbose output on standard output
  -f      specify archive file name
  -m      include manifest information from specified manifest file
  -0      store only; use no ZIP compression
  -M      do not create a manifest file for the entries
  -i      generate index information for the specified jar files
  -C      change to the specified directory and include the following file
If any file is a directory then it is processed recursively.
The manifest file name and the archive file name needs to be specified
in the same order the 'm' and 'f' flags are specified.

Example 1: to archive two class files into an archive called classes.jar:
jar cvf classes.jar Foo.class Bar.class
Example 2: use an existing manifest file 'mymanifest' and archive all the
files in the foo/ directory into 'classes.jar':
jar cvfm classes.jar mymanifest -C foo/ .

C:\trabajo\bin>_
```

Ficheros JAR

Las opciones mas comunes son:

- c: crear un fichero JAR nuevo.
- t: listar el contenido de un fichero JAR.
- x: extraer el contenido de un fichero JAR.
- f: especificar el fichero JAR (en combinación con las anteriores opciones).
- m: especificar un fichero "manifest".
- v: mostrar información del proceso por pantalla.

Ejemplo

Crear el fichero JAR:

```
Command Prompt
C:\trabajo\bin>jar cvf mijar.jar edu
added manifest
adding: edu/<in = 0> <out= 0><stored 0%>
adding: edu/upco/<in = 0> <out= 0><stored 0%>
adding: edu/upco/einf/<in = 0> <out= 0><stored 0%>
adding: edu/upco/einf/Test.class<in = 429> <out= 301><deflated 29%>
C:\trabajo\bin>
```

Listar el contenido del fichero JAR:

```
Command Prompt
C:\trabajo\bin>jar -tvf mijar.jar
 0 Thu Jan 08 21:20:04 CET 2004 META-INF/
71 Thu Jan 08 21:20:04 CET 2004 META-INF/MANIFEST.MF
 0 Thu Jan 08 20:57:16 CET 2004 edu/
 0 Thu Jan 08 20:57:16 CET 2004 edu/upco/
 0 Thu Jan 08 20:57:16 CET 2004 edu/upco/einf/
429 Thu Jan 08 21:01:52 CET 2004 edu/upco/einf/Test.class
C:\trabajo\bin>
```

Ejecución

- 1 Tanto la JVM como el compilador de Java saben buscar clases dentro de los ficheros JAR.
- 2 Para poder ejecutar aplicaciones en ficheros JAR tenemos tres opciones:
 - 3 Añadir el fichero JAR al CLASSPATH:
set CLASSPATH=c:\temp\mijar.jar
java edu.upco.einf.Test
 - 4 Añadir el fichero JAR al CLASSPATH en línea de ejecución:
java -cp c:\temp\mijar.jar edu.upco.einf.Test
 - 5 Crear un fichero JAR ejecutable.

Fichero JAR ejecutable

- 1 Los ficheros JAR contienen un fichero de descripción llamado el fichero “manifest”.
- 2 Al listar el contenido del fichero JAR en el ejemplo pudimos observar la entrada:
 - 3 META-INF/MANIFEST.MF
- 4 Si extraemos el contenido del fichero JAR y editamos el fichero “manifest” veremos algo como:
 - 5 Manifest-Version: 1.0
Created-By: 1.6.0 (Sun Microsystems Inc.)

Fichero JAR ejecutable

- Para que un fichero JAR sea ejecutable necesitamos añadir la siguiente entrada al fichero “manifest”:

- Main-Class: nombre_de_clase_principal (Nota: sin .class)*

- Main-Class: edu.upco.einf.Test

- Para añadir dicha línea al fichero “manifest”, crearemos un fichero texto con la línea y al crear el fichero JAR utilizaremos la opción `-m` para referenciarlo:

- `jar cvmf manifest.txt mijar.jar *.class`

Fichero JAR ejecutable

- Nota: es muy importante añadir un “Intro” al final de la línea “Main-Class” para que se añada correctamente.

- Para ejecutarlo, utilizaremos la opción `-jar` de la JVM:

- `java -jar mijar.jar`

- En las últimas versiones de los Sistemas Operativos: Windows y Mac, se pueden ejecutar este tipo de fichero JAR directamente pulsando dos veces sobre el fichero con el ratón.

Despliegue

- Existen distintos tipos de despliegue, dependiendo de dónde resida el código:
- Local: la aplicación reside y se ejecuta en la máquina cliente (usuario final).
- Remoto: la aplicación reside y se ejecuta en una máquina servidora.
- Mixto: la aplicación reside en una máquina servidora pero se ejecuta en la máquina cliente. O parte de la aplicación reside y se ejecuta en la máquina cliente mientras que otra parte reside y se ejecuta en la máquina servidora.

Despliegue local

- La aplicación reside y se ejecuta en la máquina cliente.
- Normalmente se distribuye a través de disco, CD, FTP, e-Mail, o cualquier otro medio en forma de uno o varios ficheros JAR.
- El usuario final necesita tener instalada la JVM (el JRE – Java Runtime Environment) para poder ejecutar código Java.

Despliegue remoto

- La aplicación reside y se ejecuta en la máquina servidora.
- Los clientes son cualquier dispositivo de acceso a la red: navegador web, móvil WAP, etc....
- Para el desarrollo de la aplicación servidora se utilizan tecnologías del Java EE como Java Servlets o Java Server Pages.

Despliegue mixto

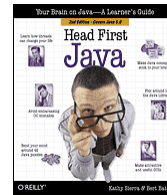
- Se trata de una solución intermedia entre el despliegue local y el despliegue remoto.
- Applets Java: las aplicaciones Java residen en un servidor y se ejecutan en un navegador (embebidas en una página HTML) en la máquina cliente.
- Java Web Start (JWS): las aplicaciones Java residen en un servidor pero se descargan en la máquina cliente a través de un navegador (normalmente) y ya quedan instaladas localmente.
- Programación distribuida: parte de la aplicación reside en la máquina cliente y parte en la máquina servidora y se comunican mediante tecnologías como: RMI, IIOP, TCP/IP.

Bibliografía



Head First Java (2nd edition)

Kathy Sierra y Bert Bates.
O'Reilly



Learning Java (2nd edition)

Patrick Niemeyer y Jonathan Knudsen.
O'Reilly.



Thinking in Java (4th edition)

Bruce Eckel.
Prentice Hall.



The Java tutorial

<http://java.sun.com/docs/books/tutorial/>