Fichiers et commande jar

Les fichiers .jar

- Les fichiers jar (java archive) sont des archives de fichiers au format zip
- Ils contiennent les .class et autres ressources nécessaires à l'exécution (icônes, etc.)
- Ils permettent de :
  - réduire la taille d'un programme + exécution
  - définir des extensions du SDK de sun
  - signer digitalement un programme
  - insérer des informations additionnelles (version, auteur)
- la manipulation des jar est un standard du SDK

La commande jar

Common JAR file operations

Operation	Command
To create a JAR file	`jar cf jar-file input-file(s)`
To view the contents of a JAR file	`jar tf jar-file`
To extract the contents of a JAR file	`jar xf jar-file`
To extract specific files from a JAR file	`jar xf jar-file archived-file(s)`
To run an application packaged as a JAR file (requires the Main-class manifest header)	`java -jar app.jar`
To invoke an applet packaged as a JAR file	`<applet code=AppletClassName.class archive="JarFileName.jar" width=width height=height> </applet>`

Utilisation basique

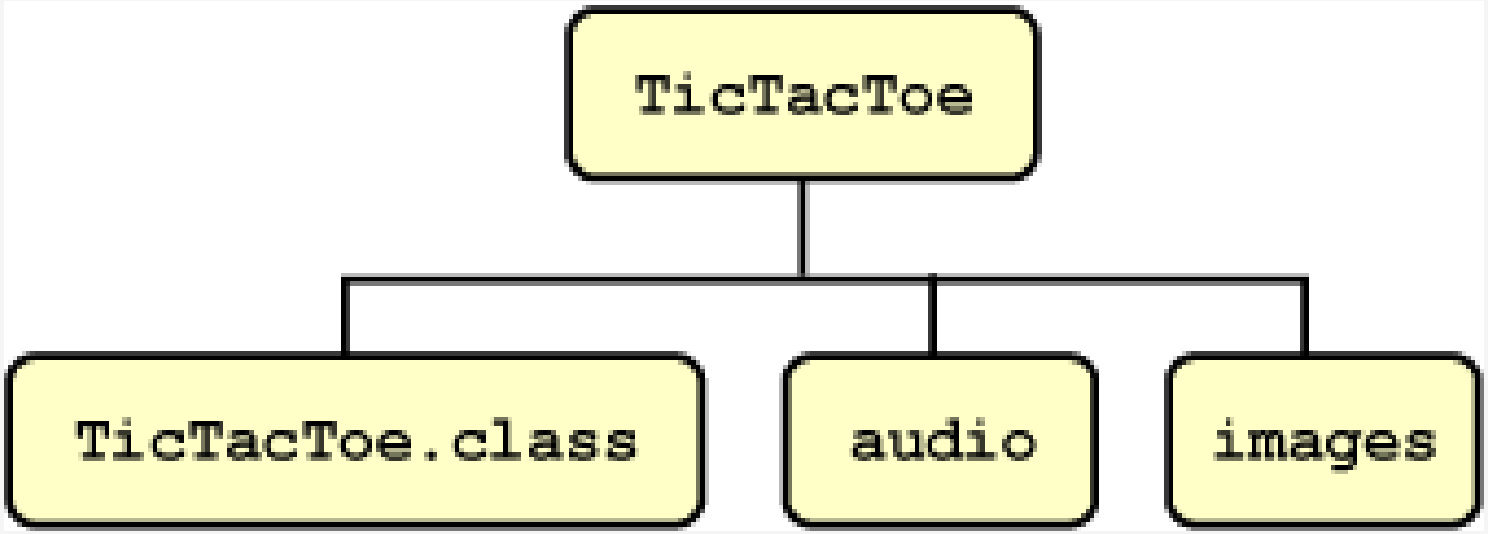
- > jar cf jar-file input-file(s)
- Les options :
  - c : création d'un fichier
  - f : messages de log dans un fichier de sortie plutôt que la console
  - jar-file : le nom du fichier jar
  - input-file(s) : les fichiers/dossiers à ajouter à l'archive (supporte le masque : « * »), les répertoires sont ajoutés avec tout leur contenu, sous-répertoires inclus

options de la commande jar

jar command options

Option	Description
`v`	Produces *verbose* output on stdout while the JAR file is being built. The verbose output tells you the name of each file as it's added to the JAR file.
`0` (zero)	Indicates that you don't want the JAR file to be compressed.
`M`	Indicates that the default manifest file should not be produced.
`m`	Used to include manifest information from an existing manifest file. The format for using this option is: `jar cmf existing-manifest jar-file input-file(s)` See Modifying a Manifest File for more information about his option. **Warning:** The manifest must end with a new line or carriage return. The last line will not be parsed properly if it does not end with a new line or carriage return.
`-C`	To change directories during execution of the command. See below for an example.

exemple



jar cvf TicTacToe.jar TicTacToe.class audio images

ou

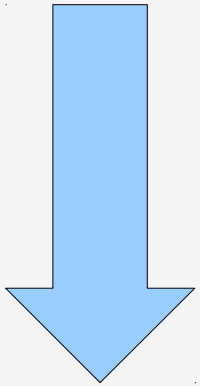
*jar cvf TicTacToe.jar **

exemple

- adding: TicTacToe.class (in=3825) (out=2222) (deflated 41%)
- adding: audio/ (in=0) (out=0) (stored 0%)
- adding: audio/beep.au (in=4032) (out=3572) (deflated 11%)
- adding: audio/ding.au (in=2566) (out=2055) (deflated 19%)
- adding: audio/return.au (in=6558) (out=4401) (deflated 32%)
- adding: audio/yahoo1.au (in=7834) (out=6985) (deflated 10%)
- adding: audio/yahoo2.au (in=7463) (out=4607) (deflated 38%)
- adding: images/ (in=0) (out=0) (stored 0%)
- adding: images/cross.gif (in=157) (out=160) (deflated -1%)
- adding: images/not.gif (in=158) (out=161) (deflated -1%)

Sans option: conservation de l'arborescence

jar cf ImageAudio.jar images audio



META-INF/MANIFEST.MF
images/cross.gif
images/not.gif
audio/beep.au
audio/ding.au
audio/return.au
audio/yahoo1.au
audio/yahoo2.au

Voir le contenu d'un jar

- > jar tf jar-file
  - option t : voir la « ***table*** of contents »
  - option f : d'un ***fichier*** particulier
- pas d'espace entre les options

jar tf TicTacToe.jar

META-INF/MANIFEST.MF
TicTacToe.class
audio/
audio/beep.au
audio/ding.au
audio/return.au
audio/yahoo1.au
audio/yahoo2.au
images/
images/cross.gif
images/not.gif

Extraire le contenu d'un jar

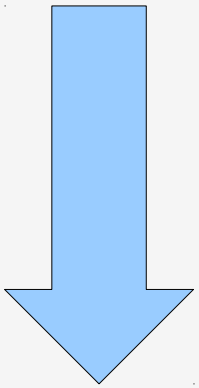
jar xf jar-file [archived-file(s)]

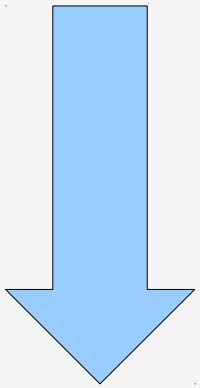
Exécution des jars

- Contexte :
  - facilité pour le web (applets)
  - simplicité d'exécution en ligne de commande
  - créer une extension du SDK

Applets

```
<applet code=TicTacToe.class
        width=120 height=120>
</applet>
```



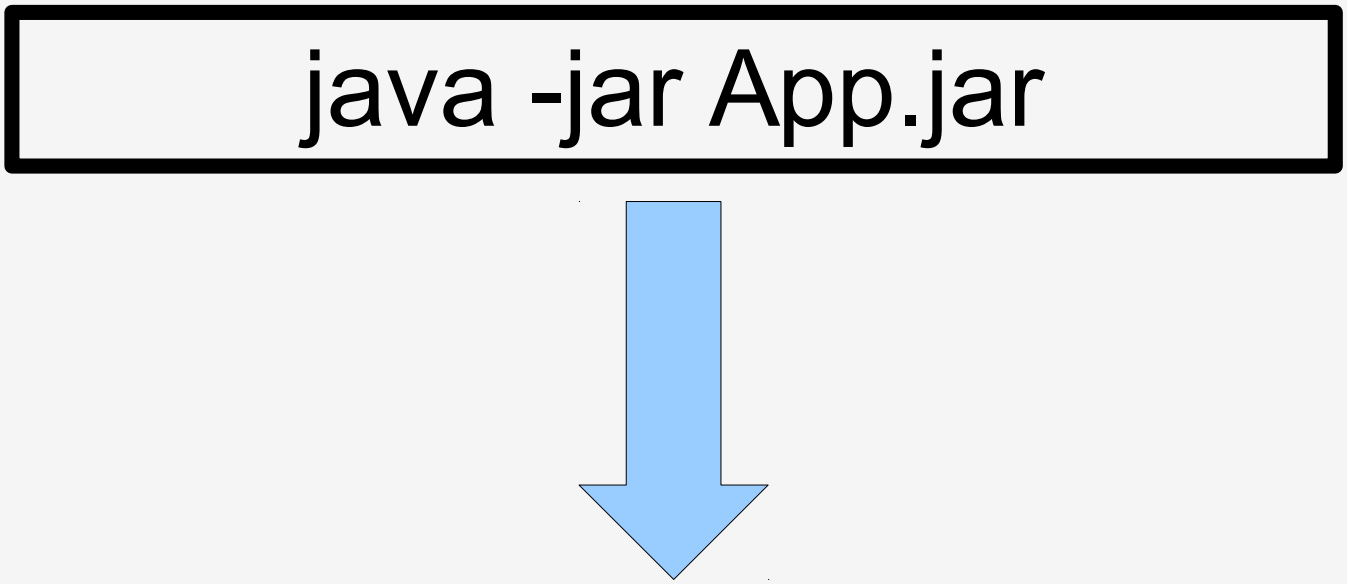


```
<applet code=TicTacToe.class
        archive="TicTacToe.jar"
        width=120 height=120>
</applet>
```

```
<applet code=TicTacToe.class
        archive="applets/TicTacToe.jar"
        width=120 height=120>
</applet>
```

Créer un jar exécutable

- Objectif :



Le fichier META-INF/MANIFEST.MF

- La commande jar crée automatiquement un fichier « manifest » ajouté à l'archive dans META-INF.

- Il contient des informations sur le contenu du fichier

Manifest-Version: 1.0
Created-By: 1.6.0 (Sun Microsystems Inc.)

Créer un jar exécutable

- Il faut inclure le point d'entrée du programme dans le fichier manifest :
  - ***Main-Class: classname***
- Comment :

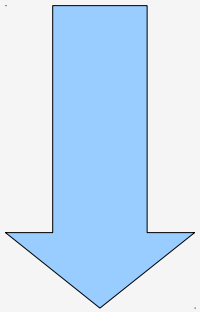
  jar cmf jar-file manifest-addition input-file(s)

- l'option **m** indique l'ajout au fichier manifest des informations contenu dans le fichier « *manifest-addition* » (contenant ***Main-Class: classname)***

exemple

- *manifest.txt*:
  - ***Main-Class: MyPackage.MyClass***
- commande jar :

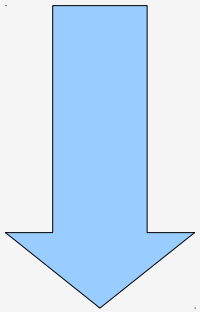
jar cfm MyJar.jar Manifest.txt MyPackage/*.class



Manifest-Version: 1.0
Created-By: 1.6.0 (Sun Microsystems Inc.)
Main-Class: MyPackage.MyClass

Ajouter des classes au classpath du jar

- Un jar peut avoir besoin d'un autre jar pour fonctionner:
  - On peut inclure cette information dans l'en-tête du fichier manifest.



```
Manifest-Version: 1.0
Class-Path: MyUtils.jar
Created-By: 1.6.0 (Sun Microsystems Inc.)
```

Exemples d'autres informations

Headers in a manifest

Header	Definition
`Name`	The name of the specification.
`Specification-Title`	The title of the specification.
`Specification-Version`	The version of the specification.
`Specification-Vendor`	The vendor of the specification.
`Implementation-Title`	The title of the implementation.
`Implementation-Version`	The build number of the implementation.
`Implementation-Vendor`	The vendor of the implementation.

Exemple de fichier manifest :

```
Name: java/util
Specification-Title: Java Utility Classes
Specification-Version: 1.2
Specification-Vendor: Sun Microsystems, Inc.
Implementation-Title: java.util
Implementation-Version: build57
Implementation-Vendor: Sun Microsystems, Inc.
```

Dans Eclipse

- File → Export → Jar file
  - Sélectionner toutes les ressources nécessaires
  - Sélectionner la classe main (dans la 3ème étape)

- « Runnable jar » file est une exportation plus complexe, nécessaire en cas de dépendance à d'autre jars dans le projet.

Récupérér des ressources depuis un fichier jar

- Java.lang.Class

getResource

```
public URL getResource(String name)
```

Finds a resource with a given name. The rules for searching resources associated with a given class are implemented by the defining class loader of the class. This method delegates to this object's class loader. If this object was loaded by the bootstrap class loader, the method delegates to ClassLoader.getSystemResource(java.lang.String).

Before delegation, an absolute resource name is constructed from the given resource name using this algorithm:

- If the `name` begins with a `'/'` (`'\u002f'`), then the absolute name of the resource is the portion of the `name` following the `'/'`.
- Otherwise, the absolute name is of the following form:

```
modified_package_name/name
```

Where the `modified_package_name` is the package name of this object with `'/'` substituted for `'.'` (`'\u002e'`).

Exemple

```
new ImageIcon(Object.class.getResource("/help.png"));
```