Java

Table des matières

1 Installation

Conseils reçus :

- utiliser le compilateur de SUN avec NetBeans.
- utiliser ANT pour les makefiles.
- utiliser Subversion ou GIT pour la gestion des versions.

Télécharger la dernière JDK chez SUN.

```
$ chmod +x jdk-6u18-linux-i586.bin
$ ./jdk-6u18-linux-i586.bin
$ alias javac="$PWD/jdk1.6.0_18/bin/javac"
$ alias java="$PWD/jdk1.6.0_18/bin/java"
$ javac -version
$ java -version
```

1.1 Debian

Télécharger NetBeans.

```
# apt-get remove gcj-4.3-base
$ ./netbeans-6.8-ml-javase-linux.sh
$ netbeans-6.8/bin/netbeans
```

1.2 Fedora

```
# yum install netbeans
```

1.3 Windows

???

1.4 Hello World

```
$ cat > hello.java
class HelloWorldApp {
    public static void main(String[] args) {
        System.out.println("Hello World!"); // Display the string.
    }
}
^D

$ javac hello.java
$ java HelloWorldApp
```

2 Netbeans

2.1 Nouveau projet

```
File > New Project > Java > Java Application

// TODO code application logic here
System.out.println("Hello World!");

shift-F11
$ java -jar dist/jpAcquisition.jar
```

2.2 Adding JDK Javadoc to NetBeans IDE

- test : cliquer sur un mot du code puis faire `ctrl-shift espace`. On peut lire : `Javadoc not found`
- Download JDK documentation from here
- In NetBeans IDE :
  1. Choose Tools > Java Platform Manager from the main window.
  2. Select the platform to which you want to add Javadoc in the left panel of the dialog box.
  3. In the Javadoc tab, click Add ZIP/Folder and specify the location of the Javadoc files.
  4. Click Close.

2.3 GUI

```
Create a JFrame container : In the Projects window, right-click the package node and choose New > JFrame Form.

public class Main {
    public static void main(String[] args) {
        new jpGUI().setVisible(true);
    }
}
```

2.4 CVS

- check out :

  ```
  In NetBeans IDE, choose Team > CVS > Checkout from the main menu. The Checkout wizard opens.
  ```

- **remarque** : CVS peut refuser de conctionner. Dans ce cas il faut le configurer comme suit :
  - CVS Root : *:ext :roche@lpnp90.in2p3.fr :/home/cvsroot*
  - use external shell : */usr/bin/ssh*

3 Bibliothèques

3.1 Socket

- serveur :

```
import java.io.*;
import java.net.*;

class UDPServer
{
   public static void main(String args[]) throws Exception
      {
         DatagramSocket serverSocket = new DatagramSocket(9876);
            byte[] receiveData = new byte[1024];
            byte[] sendData = new byte[1024];
            while(true)
               {
                  DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
                  serverSocket.receive(receivePacket);
                  String sentence = new String( receivePacket.getData());
                  System.out.println("RECEIVED: " + sentence);
                  InetAddress IPAddress = receivePacket.getAddress();
                  int port = receivePacket.getPort();
                  String capitalizedSentence = sentence.toUpperCase();
                  sendData = capitalizedSentence.getBytes();
                  DatagramPacket sendPacket =
                  new DatagramPacket(sendData, sendData.length, IPAddress, port);
                  serverSocket.send(sendPacket);
               }
      }
}
```

- client :

```
import java.io.*;
import java.net.*;

class UDPClient
{
   public static void main(String args[]) throws Exception
   {
      BufferedReader inFromUser =
```

```
            new BufferedReader(new InputStreamReader(System.in));
        DatagramSocket clientSocket = new DatagramSocket();
        InetAddress IPAddress = InetAddress.getByName("localhost");
        byte[] sendData = new byte[1024];
        byte[] receiveData = new byte[1024];
        String sentence = inFromUser.readLine();
        sendData = sentence.getBytes();
        DatagramPacket sendPacket = new DatagramPacket(sendData, sendData.length, IPAddress, 9876);
        clientSocket.send(sendPacket);
        DatagramPacket receivePacket = new DatagramPacket(receiveData, receiveData.length);
        clientSocket.receive(receivePacket);
        String modifiedSentence = new String(receivePacket.getData());
        System.out.println("FROM SERVER:" + modifiedSentence);
        clientSocket.close();
    }
}
```

- tests :

```
$ nc -u -l -p 9876 localhost
$ nc -u localhost 9876
$ javac *.java
$ java UDPServer
$ java UDPClient
```

- bug : `Network unreachable in java only`

```
This is a symptom of a bug in sun's java6 regarding ipv6.

Try
$ /sbin/sysctl net.ipv6.bindv6only

If this value is 1 you need to disable the setting:

$ sudo /sbin/sysctl net.ipv6.bindv6only=0

For a persistent fix:
Check /etc/sysctl.conf and /etc/sysctl.d/* for this setting and restart procfs after changing it.

If this was your problem, you should have networking back :)

Cheers.
```

- ping : status is true if the machine is reachable by ping ; false otherwise.

```
String host = "172.16.0.2"
int timeOut = 3000; // I recommend 3 seconds at least
boolean status = InetAddress.getByName(host).isReachable(timeOut)
```

3.2 JFreeChart

1. Download the latest version of the JCommon class library, and the latest version of the JFreeChart class library

2. Unpack it to a directory on your computer (almost anywhere is fine).

3. From the ant subdirectory of the just-unpacked JCommon and JFreeChart, run ant javadoc to generate the Javadocs locally. You can skip this step, but then you'll be missing the API documentation.
4. In NetBeans, select the Library Manager item from the Tools menu
5. Click on the *New Library...* button and enter *JCommon-1.0.16* as the library name.
6. With the Classpath tab selected, click on the *Add JAR/Folder...* button and select the *jcommon-1.0.16.jar* file from the JCommon directory created back in step 1.
7. With the Sources tab selected, click on the *Add JAR/Folder...* button and select the source directory for JCommon.
8. With the Javadoc tab selected, click on the *Add ZIP/Folder...* button and select the javadoc directory for JCommon (refer to step 2).
9. Click on the *New Library...* button and enter *JFreeChart-1.0.13* as the library name.
10. Add the *lib/jfreechart-1.0.13.jar* file, the source directory and the javadoc directory as done above.

At this point, you have complete the configuration of the libraries.

The next section shows how to create a new project in NetBeans that depends on these libraries.

1. In the Projects pane, you'll see a Libraries node in the project. Right-click o node, select *Add Library...* and select the JFreeChart and JCommon libraries.
2. NetBeans has already created a Main.java source file copy and paste the follo code into the main method of this source file bellow.
3. Select *Fix Imports* from the Source menu, then compile and run the application. Notice how you can browse the JFreeChart/JCommon source files and step through the code while debugging.

That's all there is to it !

```
public static void main(String[] args) {
    // create a dataset...
    DefaultPieDataset data = new DefaultPieDataset();
    data.setValue("Category 1", 43.2);
    data.setValue("Category 2", 27.9);
    data.setValue("Category 3", 79.5);
    // create a chart...
    JFreeChart chart = ChartFactory.createPieChart(
        "Sample Pie Chart",
        data,
        true,     // legend?
        true,     // tooltips?
        false     // URLs?
    );
    // create and display a frame...
    ChartFrame frame = new ChartFrame("First", chart);
    frame.pack();
    frame.setVisible(true);
}
```

Pour tester en ligne de commande :

```
$ java -jar dist/jpAcquisition.jar dist/lib/jcommon-1.0.16.jar dist/lib/jfreechart-1.0.13.jar

ou plus simple

$ cd dist
$ java -jar jpAcquisition.jar
```

3.3 XML

```
JDOM is an open source Java-based document object model for XML that was designed specifically for the Java platform
so that it can take advantage of its language features.
JDOM integrates with Document Object Model (DOM) and Simple API for XML (SAX), supports XPath and XSLT.
```

Télécharger le fichier *jdom.jar* JDom puis l'insérer dans NetBeans comme on l'a détaillé ci-dessus pour JFreeChart. Insérez également le fichier *jaxen.jar* dont on a besoin pour utiliser xpath.

fichiet *foo.xml* :

```
<shop name="shop for geeks" location="Tokyo, Japan">
  <computer name="iBook" price="1200$" />
  <comic_book name="Dragon Ball vol 1" price="9$" />
  <geekyness_of_shop price="priceless" />
</shop>
```

4 How-to

4.1 Documentation des api

La documentation est disponible en ligne .

4.2 Packaging de l'exécutable

```
$ cd nectar
$ ant dist                      # compilation (marche ssi netbeans est installé avec les librairies)
$ java -jar dist/nectar.jar     # test
$ tar -zcf nectar.tgz dist/     # archivage
```

Remarque : Pour ajouter le fichier de configuration xml au répertoire *dist* il faut ajouter la cible suivante au fichier *build.xml* :

```
    <target name="-post-jar">
        <copy file="nectar.xml" flatten="true" todir="${dist.dir}"/>
    </target>
```

4.3 Tests unitaires

Fichier *truc.java*

```
package monPaquet;

public class truc {

    public static void main(String[] args) {
        // TODO unitary test code here
    }
```

Lancement de la section main de la classe **truc** (fichier *truc.java*) :

```
$ java -cp dist/lib/jcommon-1.0.16.jar:dist/lib/jfreechart-1.0.13.jar:dist/lib/jdom.jar:dist/monPaquet.jar monPaquet.truc
```

```
ou encore
$ java -cp dist/monPaquet.jar:‘find dist/lib/ -name ’*.jar’ -exec echo -n "{}:" \;‘ monPaquet.truc
```

Remarque, ces 2 lignes sont équivalentes :

```
$ java -cp dist/monPaquet.jar:‘find dist/lib/ -name ’*.jar’ -exec echo -n "{}:" \;‘ MonPaquet.Main
$ java -jar dist/monPaquet.jar
```

Lancement sous windows (attention ça risque d'effacer la route par défaut) :

```
> route -p ADD 192.168.1.0 MASK 255.255.255.0 134.158.152.165
```

4.4 Applet

Configuration de FIREFOX (cf ici) :

```
$ cd ~/.mozilla/plugins
$ ln -s /home/nroche/.../jre1.6.0_18/plugin/i386/ns7/libjavaplugin_oji.so
```

Redémarez FIREFOX et vérifié la bonne installation du plugin via l'adresse about :plugins
remarque : Glop marche maintenant !