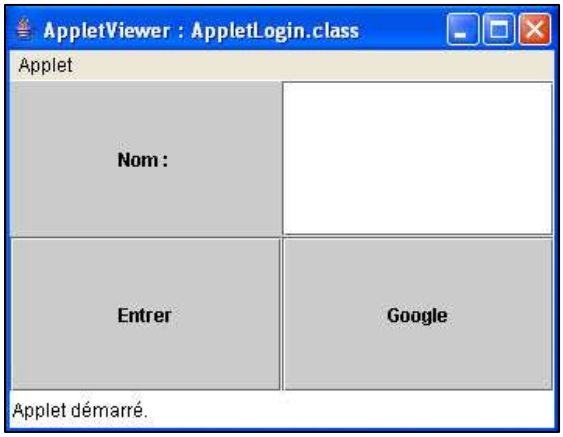
Quelques notes pratiques sur les applets élémentaires avec Sun ONE Studio et Tomcat



1. Une applet basique

1) Sun ONE Studio : projet AppletDemo
- construction d'une applet AppletLogin :

AppletLogin.java

```
/*
 * AppletLogin.java
 *
 */

/**
 * @author  Claude
 */
import javax.swing.*;
import java.net.*;

public class AppletLogin extends javax.swing.JApplet
{

    public AppletLogin() { initComponents();  }

    public void start()
    {
        System.out.println("(Re)démarrage de l'applet ...");
    }

    public void init()
    {
        System.out.println("Initialisation de l'applet ...");
    }

    private void initComponents() {…}

    private javax.swing.JButton jButton2;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton jButton1;
    private javax.swing.JTextField ZTNom;
}
```

2) Exécuter avec le **viewer d'applet** : voir aussi les affichages dans la console et la page html générée :

```
<HTML>
<HEAD>
  <TITLE>Applet HTML Page</TITLE>
</HEAD>
<BODY>

<H3><HR WIDTH="100%">Applet HTML Page<HR WIDTH="100%"></H3>

<P>
<APPLET code="AppletLogin.class" width=350 height=200></APPLET>
</P>

<HR WIDTH="100%"><FONT SIZE=-1><I>Generated by NetBeans IDE</I></FONT>
</BODY>
</HTML>
```

2. Construire sa propre page HTML

Construire une autre page HTML dans Sun ONE :

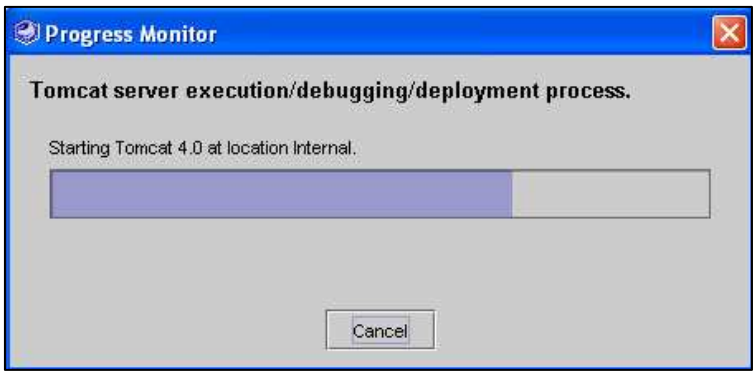
ShopLogin.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
 <HEAD>
  <TITLE></TITLE>
 </HEAD>
 <BODY>

 <H3>Bienvenue dans notre magasin virtuel !!!</H3>

 <APPLET code="AppletLogin.class" width=350 height=200></APPLET>

 </BODY>
</HTML>
```

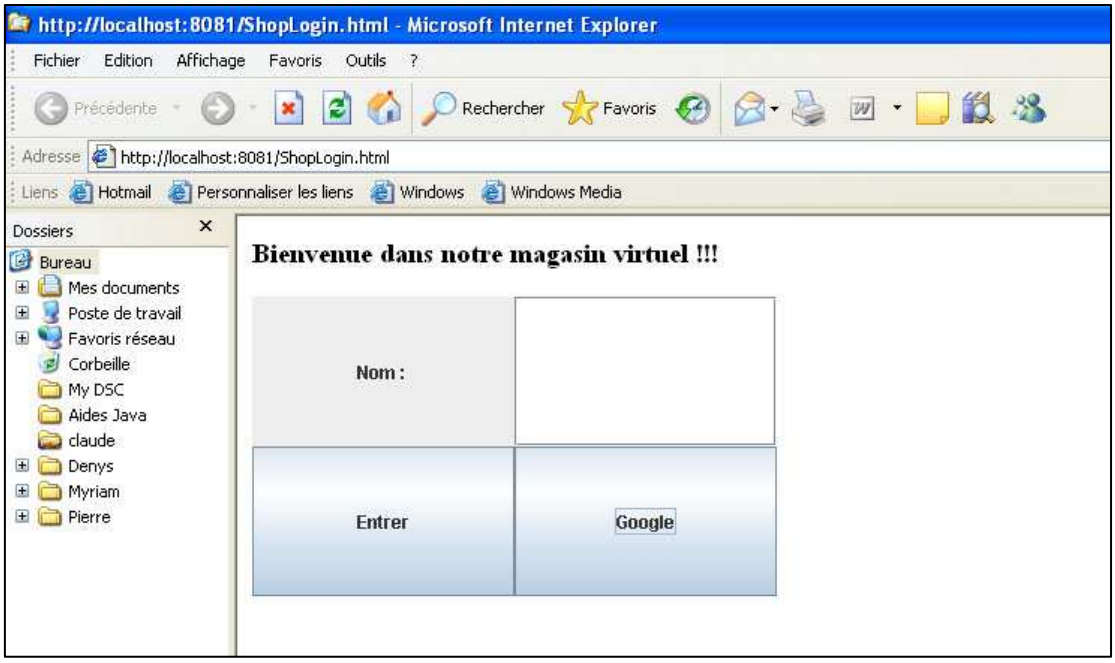
"Exécuter" cette page :

a) l'IDE demande à construire un "module Web" (une configuration particulière de répertoires) :

b) puis lance un Tomcat intégré :



qui nous conduit dans un browser (ici, Internet Explorer) :



A remarquer la console Java (Outils → Console Java (Sun)) :

3. Quelques précisions techniques

1) Tomcat est un "moteur à servlets et JSP" qui peut aussi fonctionner comme serveur Web; il attend sur le port 8083 (par exemple – tout dépend comment on a installé Sun ONE) comme on peut le voir avec :

```
C:\Documents and Settings\Claude>netstat -an | find "80"
…
  TCP    0.0.0.0:8081           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8083           0.0.0.0:0              LISTENING
```

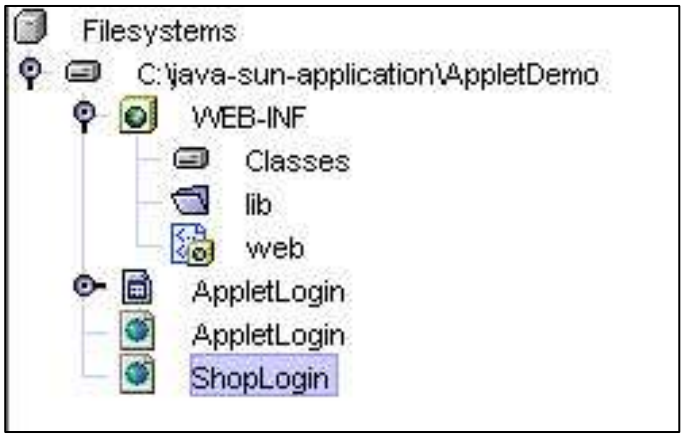
ou dans Tools → Options → IDE Configuration → Server and External Tools Settings → HTTP Server :

Hosts with Granted Access	Selected Hosts:
Port	8083
Running	True

2) Pour que Sun ONE utilise browser externe (au lieu d'un browser interne écrit en Swing), il faut configurer par Tools → Options :

2.1) → Debugging and Executing → JSP and servlets settings → Web browser : External browser (Windows)
2.2) → Debugging and Executing → IDE Configuration → Server and External Tools Settings → External Browser (Windows) → DDE Server : IEXPLORE

3) La structure de répertoires créée en plus :

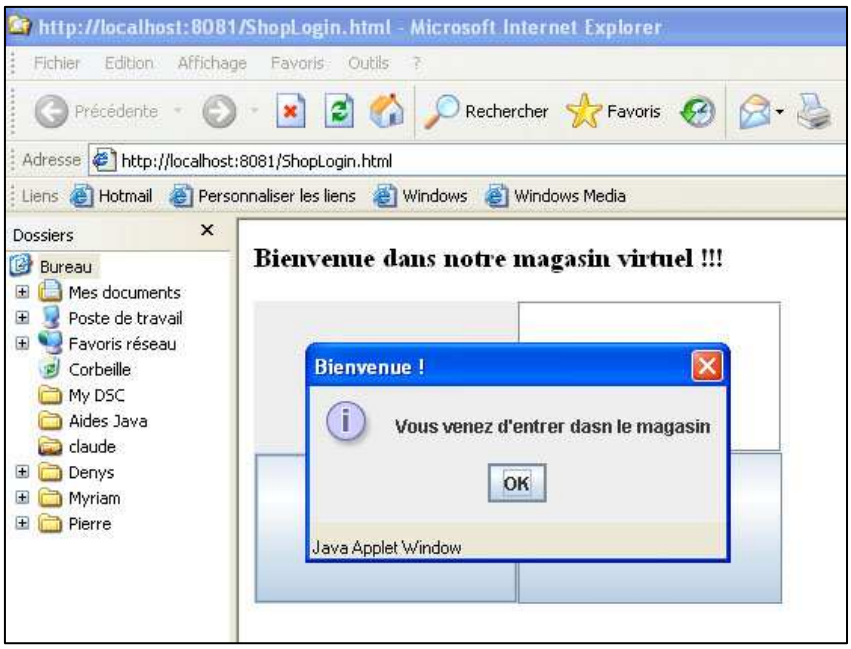


4. La gestion d'un événement

On ajoute l'apparition d'une nboîte de dialogue si on appuie sur Entrer :

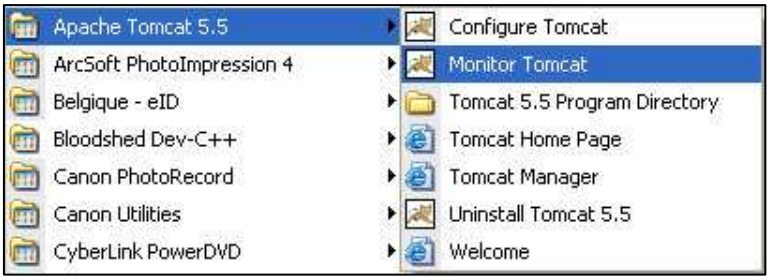
```
   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      System.out.println("On a appuyé sur Entrer :-) ");       // Add your handling code here:
      JOptionPane.showMessageDialog(this, "Vous venez d'entrer dasn le magasin",
"Bienvenue !", JOptionPane.INFORMATION_MESSAGE,null);
   }
```

Résultat :



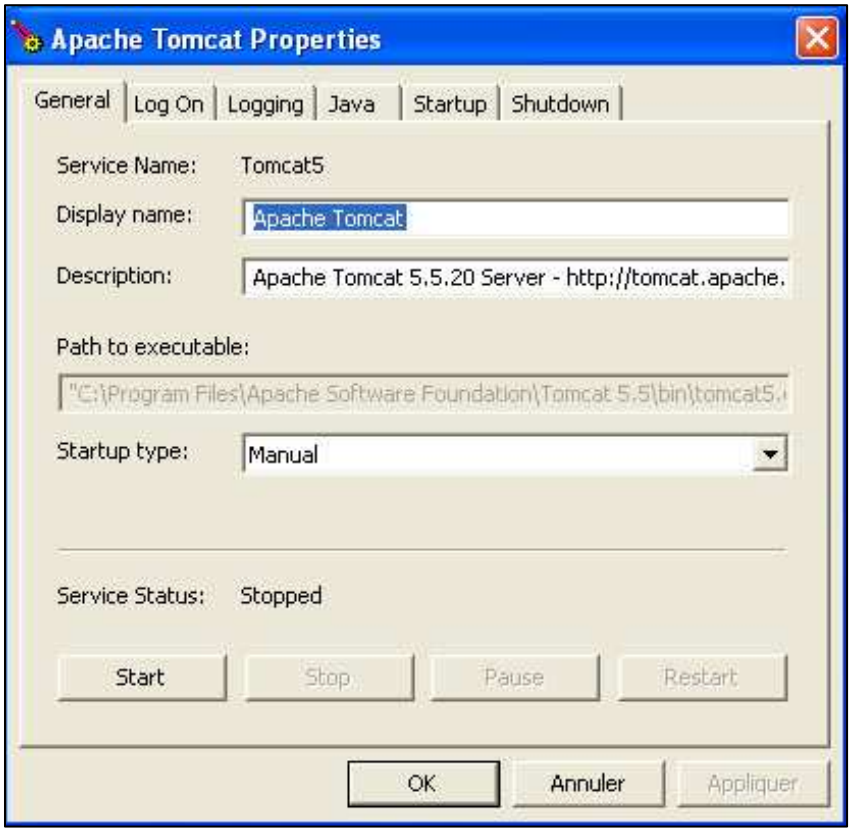
5. Tomcat en tant que serveur autonome

On peut lancer Tomcat en tant que serveur indépendant (non intégré) :

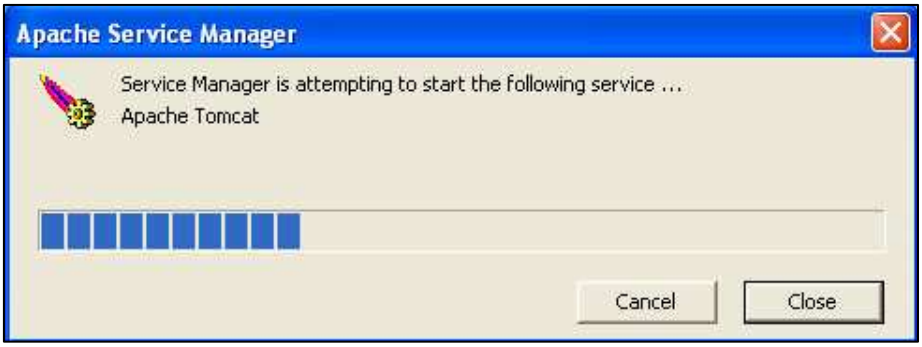


Résultat discret : une icône dans la barre d'état

Un double clic sur cette icône ➔



On peut donc faire effectivement démarrer Tomcat :



On peut vérifier que le serveur attend sur le port 8080 (ou autre, du moment que ce port n'est pas pris par un autre serveur, comme le Tomcat intégré de Sun ONE) :

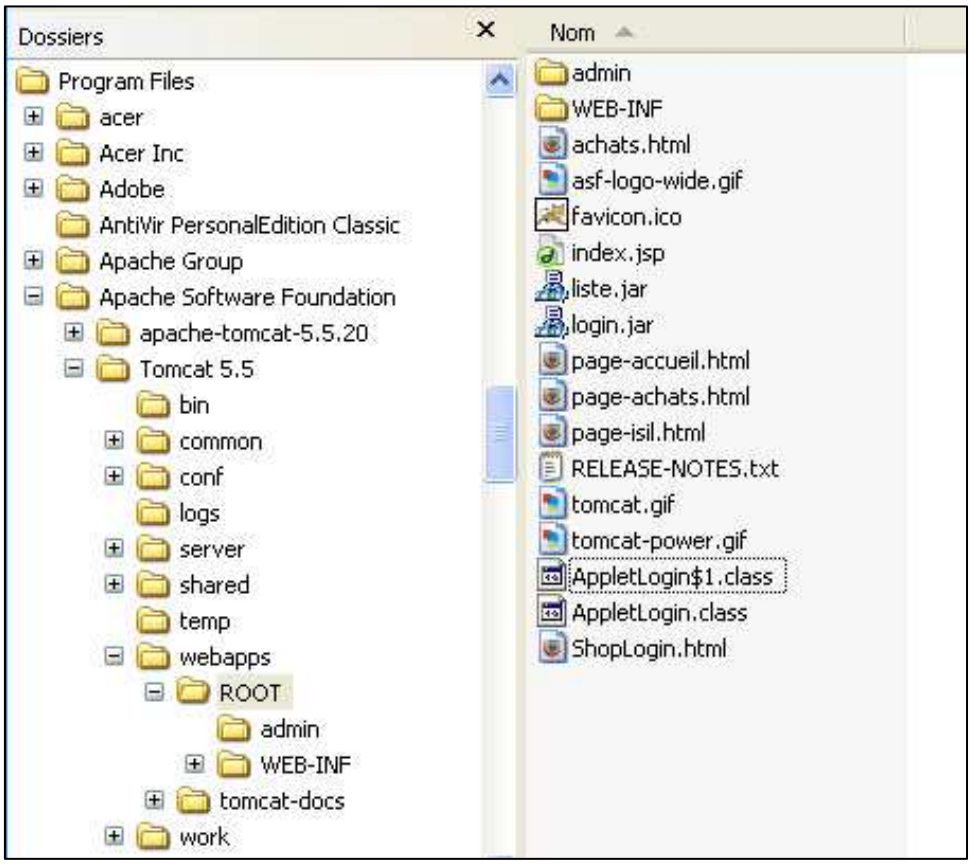
```
C:\Documents and Settings\Claude>netstat -an | find "80"
…
  TCP    0.0.0.0:8009           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8080           0.0.0.0:0              LISTENING
  TCP    0.0.0.0:8083           0.0.0.0:0              LISTENING
…
```

6. Utiliser l'applet avec Tomcat autonome

Tomcat est configuré de manière à trouver ses ressources à partir du répertoire :

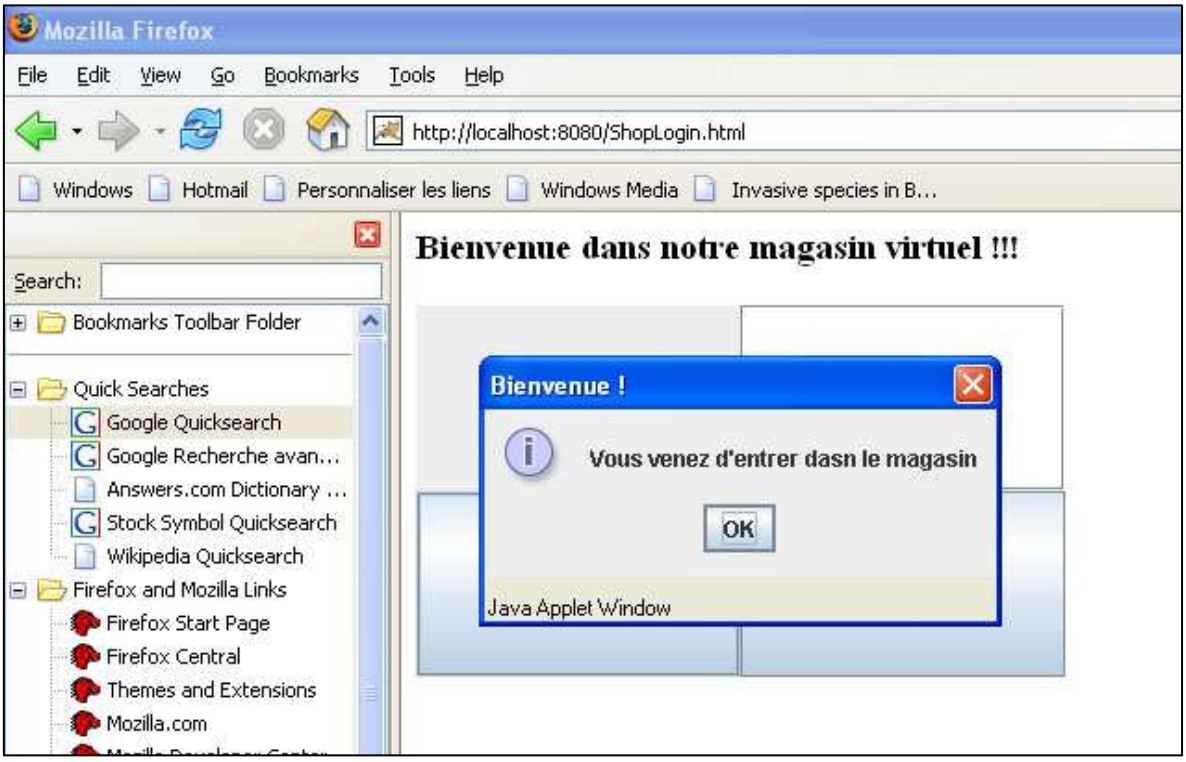
C:\Program Files\Apache Software Foundation\Tomcat 5.5\webapps\ROOT

C'est donc là que nous allons placer la page HTML et les fichiers class de l'applet :

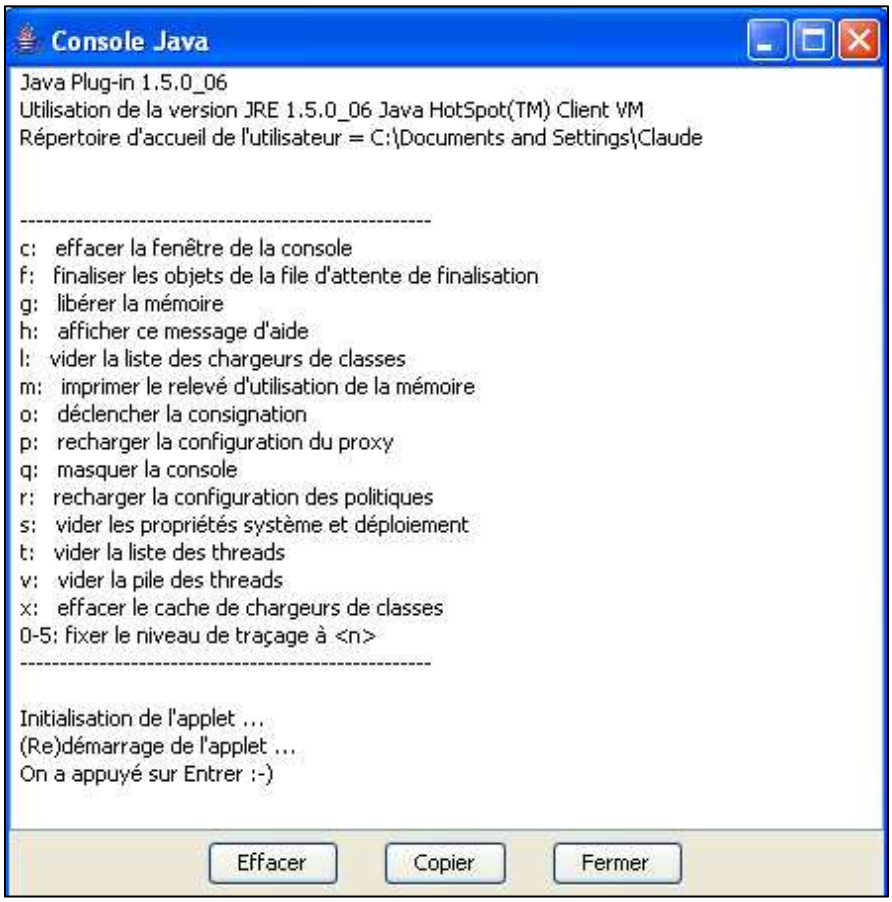


On peut alors charger la page depuis un browser quelconque par :

http://localhost:8080/ShopLogin.html



avec dans la console Java :



Remarque
Si la console Java n'est pas disponible dans Firefox (elle n'est pas installée par défaut), on peut l'intégrer en allant sur le site des addons de Firefox :

https://addons.mozilla.org/en-US/firefox/addon/141

8. L'utilisation des jars

Il est évidemment un peu lourd de devoir recopier les fichiers class dans leurs répertoires respectifs : il est bien plus simple d'utiliser un jar. Donc, après ajustement du PATH au moyen d'un fichier java-env.bat :

```
cd \java-sun-application
set PATH=c:\j2sdk1.4.2_03\bin;%PATH%
java -version
```

```
C:\java-sun-application>java-env
    C:\java-sun-application\AppletDemo>cd \java-sun-application
    C:\java-sun-application>set PATH=c:\j2sdk1.4.2_03\bin;C:\Program Files\Reflectio
    n;C:\WINDOWS\system32;C:\WINDOWS;C:\WINDOWS\System32\Wbem;C:\Program
    Files\ATI Technologies\ATI Control
    Panel;C:\Gemplus\GemXpresso.rad3\bin;C:\Gemplus\GemXpres
    so.rad3\bin;;C:\PROGRA~1\FICHIE~1\MUVEET~1\030625
    C:\java-sun-application>java -version
    java version "1.4.2_03"
    Java(TM) 2 Runtime Environment, Standard Edition (build 1.4.2_03-b02)
    Java HotSpot(TM) Client VM (build 1.4.2_03-b02, mixed mode)
```

on peut alors créer le jar :

```
C:\java-sun-application\AppletDemo>jar cvf shoplogin.jar *.class
manifest ajoutÚ
ajout : AppletLogin$1.class(entrÚe = 1694) (sortie = 1535)(9% compressÚs)
ajout : AppletLogin.class(entrÚe = 2287) (sortie = 1223)(46% compressÚs)
```

Créons une page html qui va utiliser ce jar :

ShopLoginJar.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<HTML>
 <HEAD>
  <TITLE></TITLE>
 </HEAD>
 <BODY>
 <H3>Bienvenue dans notre magasin virtuel avec jar !!!</H3>

 <APPLET code="AppletLogin.class" archive="shoplogin.jar" width=350 height=200></APPLET>


 </BODY>
</HTML>
```

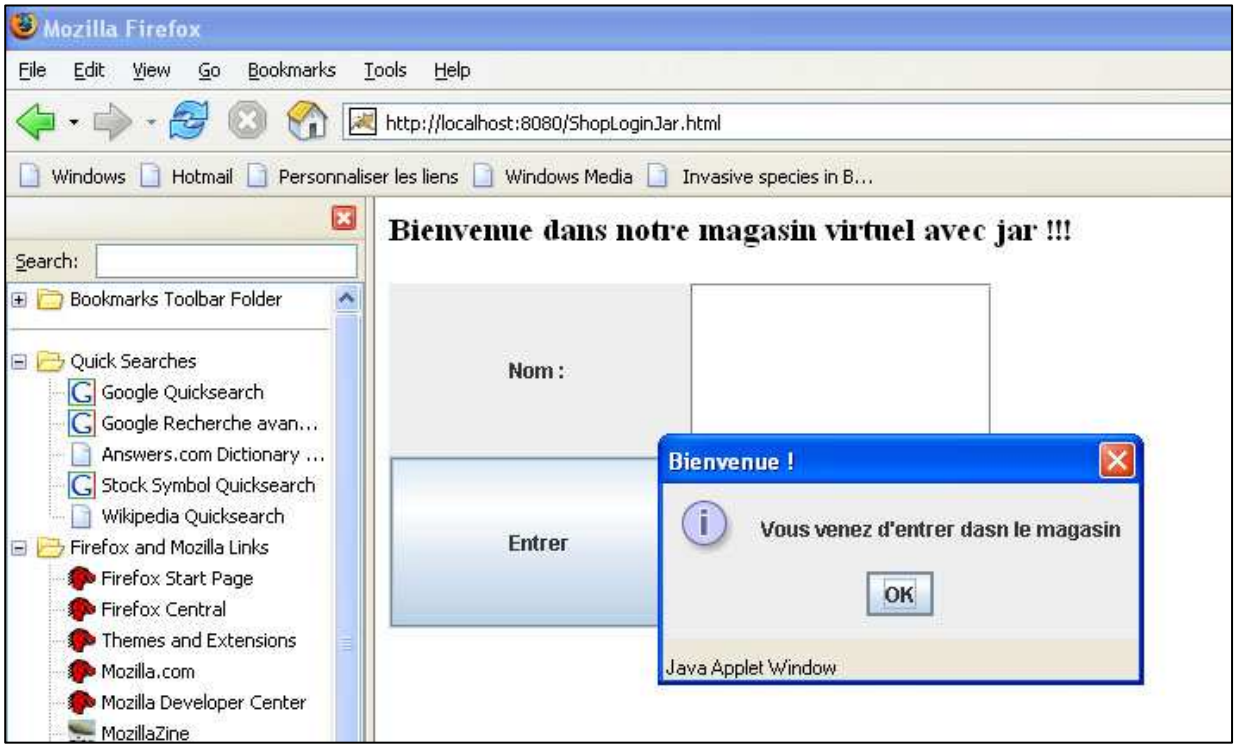
On transporte la page et le jar dans le répertoire de Tomcat :

C:\Program Files\Apache Software Foundation\Tomcat 5.5**webapps\ROOT**

puis on invoque la page depuis un browser :

http://localhost:8080/ShopLoginJar.html

ce qui donne :



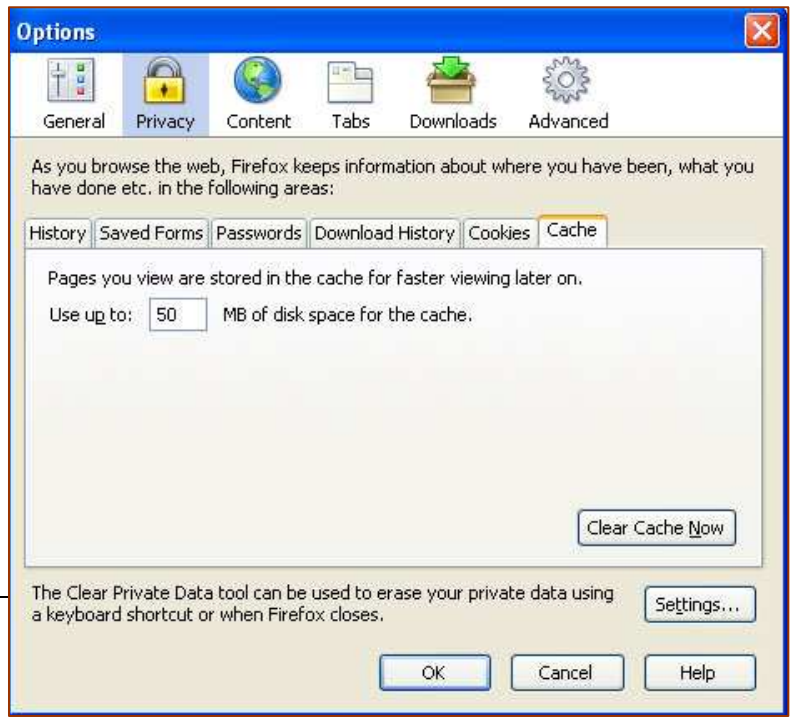
9. Le piège des caches

Les développements Web sont fréquemment handicapés par l'utilisation de caches qui font que la dernière version de la page, du jar ou d'une classe Java ne sont pas pris en compte. On veillera donc à vider ces caches :

- **au niveau du browser utilisé**;

par exemple, pour Firefox

Tools → Options ➔



ou pour Internet Explorer :

Outils → Options Internet → Général : fichiers Internet temporaires

♦ **au niveau du chargeur de classe Java** qui utilise lui aussi un cache : on utilise la commande de la console Java :

x: effacer le cache de chargeurs de classes

10. Le transfert vers une autre page

Supposons à présent que si aucun nom n'est entré dans le GUI de l'applet, on veuille rediriger l'utilisateur sur une page d'erreur. Ceci est possible

1) en se procurant le "contexte de l'applet" (en clair, un objet qui référencie le browser qui affiche la page HTML hôte) par :

public **AppletContext getAppletContext**()

2) en utilisant l'objet ainsi récupéré pour appeler sa méthode :

public abstract void **showDocument**(URL url)

qui transfert l'exécution vers la ressource Web (donc ici une page) dont l'URL est spécifié comme paramètre (elle lance une exception MalformedURLException sir l'URL est mal formée).

On peut donc imaginer comme réponse à l'appui sur le bouton "Entrer" :

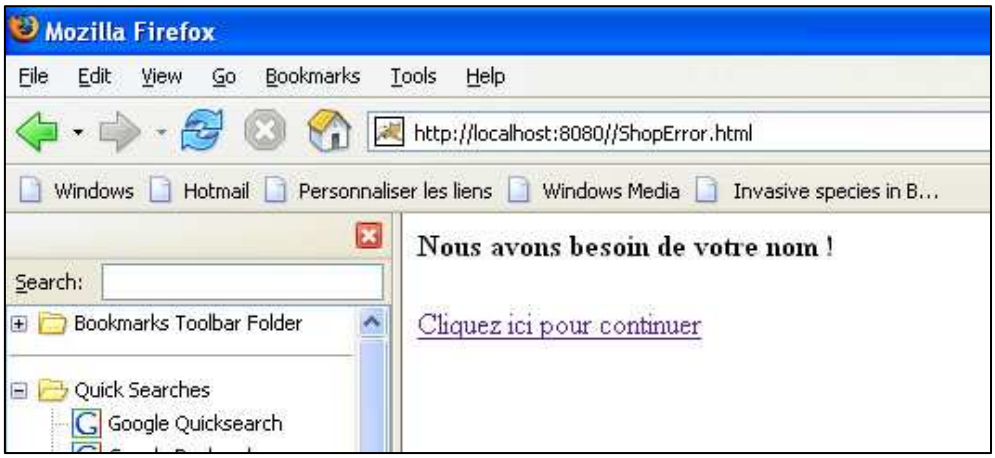
```
   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
      System.out.println("On a appuyé sur Entrer :-) ");
      String nomEntre = ZTNom.getText();
      if (!nomEntre.equals(""))
         JOptionPane.showMessageDialog(this, "Vous venez d'entrer dasn le magasin",
"Bienvenue !", JOptionPane.INFORMATION_MESSAGE,null);
      else
      try
      {
         getAppletContext().showDocument(
                  new URL("http://localhost:8080//ShopError.html"));
      }
      catch (MalformedURLException e)
      {
         System.out.println("Erreur d'URL : " + e.getMessage());
      }
   }
```

avec une page d'erreur de ce type :

ShopError.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
 <HEAD>
  <TITLE></TITLE>
 </HEAD>
 <BODY>
 <H4>Nous avons besoin de votre nom !</H4>

 <P> <A href="ShopLoginJar.html">Cliquez ici pour continuer</A>
 </BODY>
</HTML>
```

Une nouvelle exécution avec un nom vide donne effectivement :



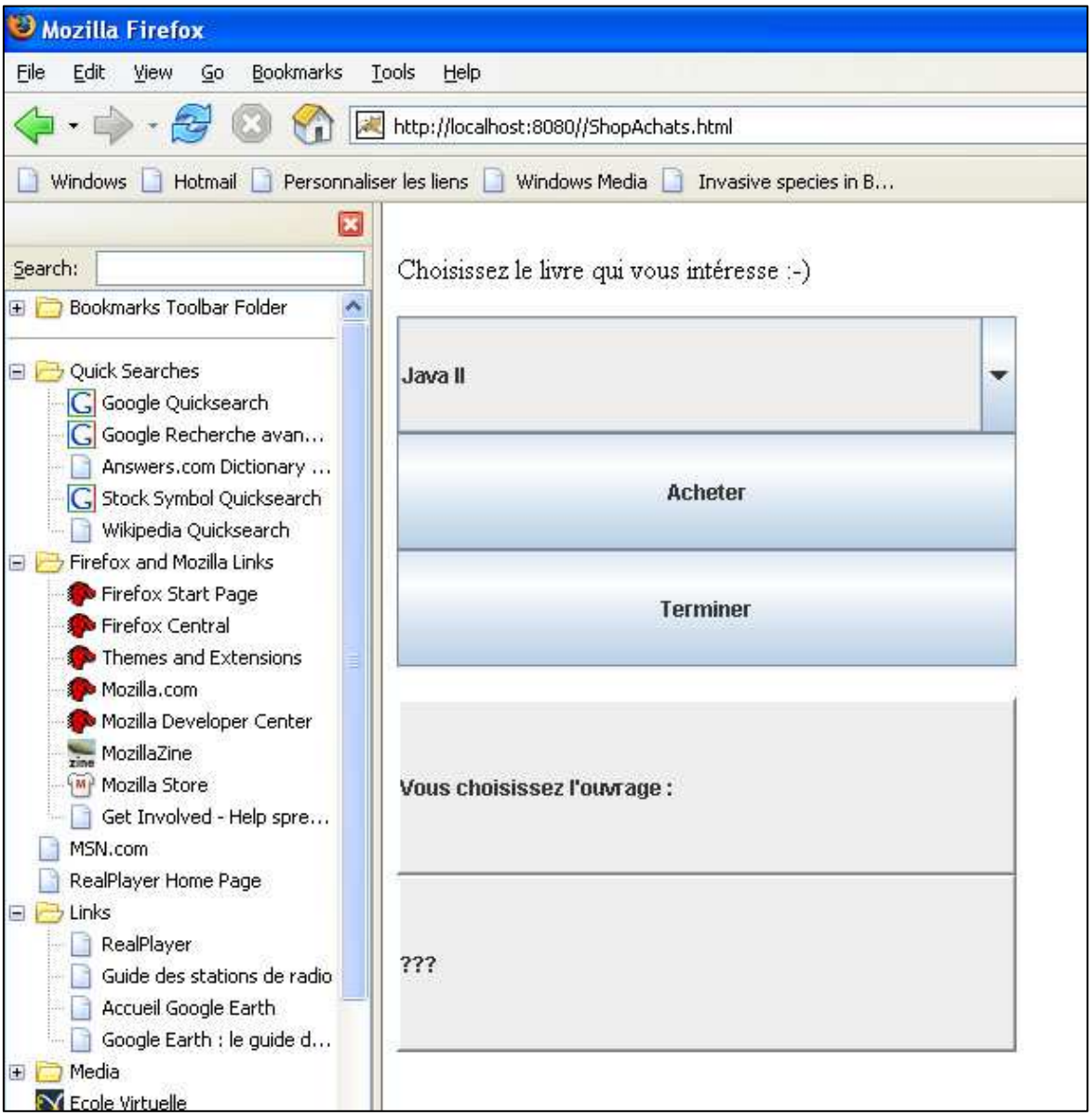
11. La communication entre deux applets d'une même page

Supposons à présent que l'entré d'un nom valide conduit à une nouvelle page HTML vouée à l'achat d'un livre (par exemple) – la page se somme ShopAchats :

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
        System.out.println("On a appuyé sur Entrer :-) ");
        String nomEntre = ZTNom.getText();
        if (!nomEntre.equals(""))
        {
            JOptionPane.showMessageDialog(this, "Vous venez d'entrer dans le magasin",
"Bienvenue !", JOptionPane.INFORMATION_MESSAGE,null);
            try
            {
                getAppletContext().showDocument(
                    new URL("http://localhost:8080//ShopAchats.html"));
            }
            catch (MalformedURLException e)
            {
                System.out.println("Erreur d'URL : " + e.getMessage());
            }
        }
        else
```

```
        try
        {
           getAppletContext().showDocument(
                    new URL("http://localhost:8080//ShopError.html"));
        }
        catch (MalformedURLException e)
        {
           System.out.println("Erreur d'URL : " + e.getMessage());
        }
    }
```

Cette page présente la caractéristique de faire intervenir deux applets :



ainsi que le montre son code HTML :

ShopAchats.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
  <HEAD>
    <TITLE></TITLE>
  </HEAD>
  <BODY>
<br>Choisissez le livre qui vous intéresse :-)<p>
<P>

<APPLET code="AppletListe.class" archive="liste.jar" name="liste" width=350
height=200></APPLET>

</P>
<APPLET code="AppletVisualisation.class" archive="liste.jar" name="visualise"
width=350 height=200></APPLET>

  </BODY>
</HTML>
```

Nous souhaiterions que l'appui sur le bouton Acheter de la première applet (appelée AppletListe) provoque l'apparition du livre choisi dans le la zone d'affichage inférieure de la deuxième applet (appelée AppletVisualisation).

Le problème est donc clairement de **faire communiquer les deux applets**. Pour cela, chaque applet doit posséder retrouver les références des autres applets de la même page HTML. Ceci est possible grâce aux méthodes de l'AppletContext :

public Enumeration **getApplets**()
public Applet **getApplet**(String name)

Dans ce dernier cas, le nom "name" désigne le nom donné à l'applet dans la page HTML au moyen de l'attribut "name" du tag <applet> (donc dans notre cas "liste" et "visualise"). Si donc la deuxième applet possède une méthode setNomPieceChoisie() qui modifie le texte affiché dans la zone de texte visée :

AppletVisualisation.java

```
/*
 * AppletVisualisation.java
 *
 */

/**
 * @author  Claude
 */

public class AppletVisualisation extends javax.swing.JApplet
{
    private String nomPieceChoisie;

    public AppletVisualisation() {  initComponents();  }
```

```
    private void initComponents() {…}

    public void setNomPieceChoisie(String np)
    {
       nomPieceChoisie = np;
       System.out.println("Reception du nom choisi : " + nomPieceChoisie);
       LOuvrageChoisi.setText(nomPieceChoisie);
    }

  private javax.swing.JLabel jLabel1;
  private javax.swing.JLabel LOuvrageChoisi;
}
```

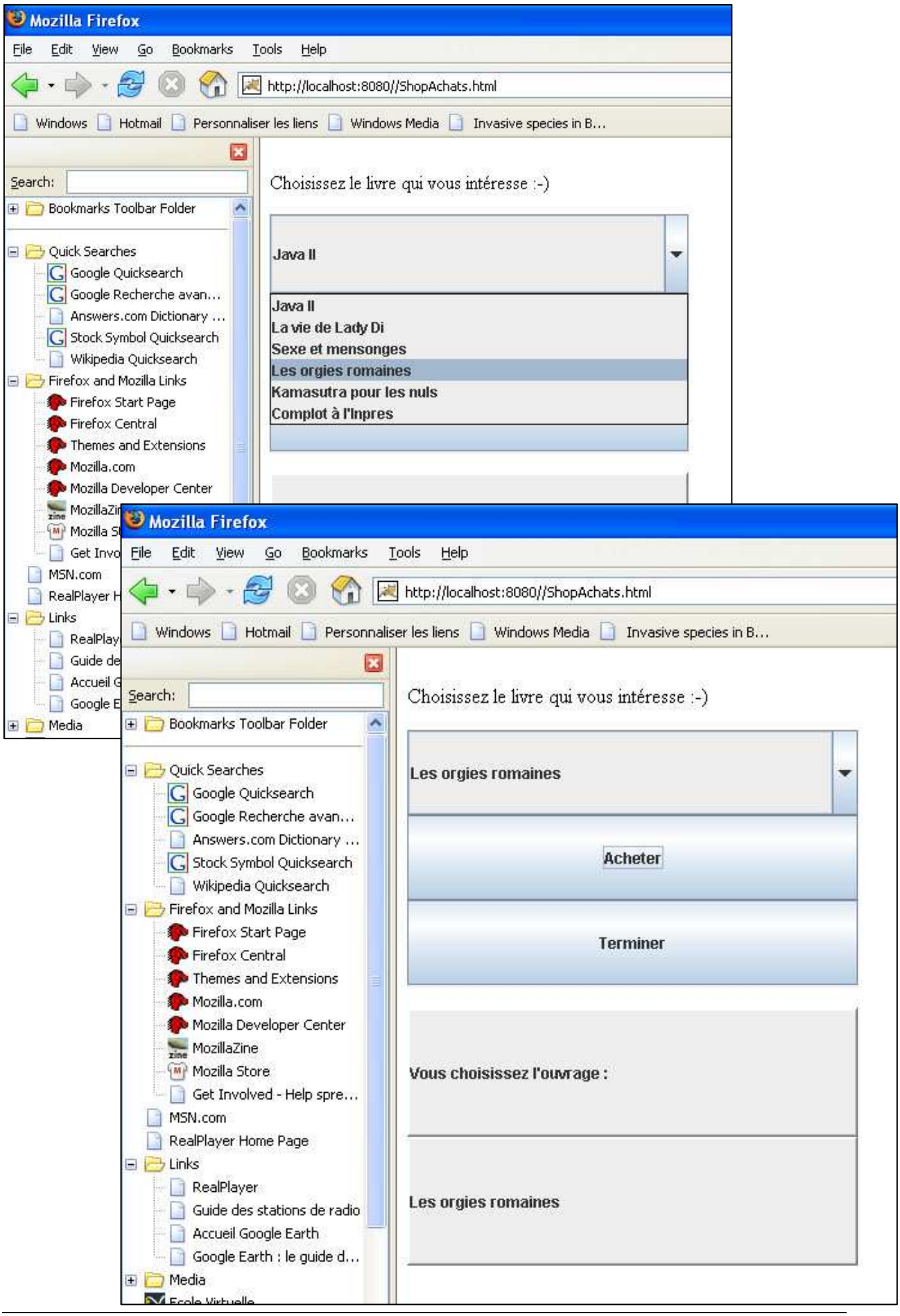
la première applet, outre le fait qu'elle propose la liste des livres en vente, n'a plus qu'à rechercher la référence de l'applet AppletVisualisation qui possède cette méthode :

AppletListe.java

```
/*
 * AppletListe.java
 *
 */

/**
 * @author  Claude
 */
import java.util.*;
import javax.swing.*;

public class AppletListe extends javax.swing.JApplet
{
  private String[] nomsPieces =
  {   "Java II", "La vie de Lady Di", "Sexe et mensonges", "Les orgies romaines",
      "Kamasutra pour les nuls", "Complot à l'Inpres" };

  public AppletListe()
  {
     initComponents();
     for (int i=0; i<nomsPieces.length; i++)
     {
        CBListe.addItem(nomsPieces[i]);
     }
  }

  private void initComponents() {…}

  private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
  {
     String np = (String)CBListe.getSelectedItem();
     System.out.println("Pièce cherchée : " + np);
```

```
        System.out.println("Début de recherche");
        for (Enumeration e = getAppletContext().getApplets();e.hasMoreElements();)
        {
           try
           {
              JApplet t = (JApplet) e.nextElement();
              System.out.println("Applet trouvée : " + t.getParameter("name"));
           }
           catch (ClassCastException err)
           {
              System.out.println("Erreur de casting : " + err.getMessage());
           }
        }

        JApplet friend = (JApplet)getAppletContext().getApplet("visualise");
        if (friend!=null) System.out.println("Applet visualise trouvée");
        else System.out.println("Applet visualise PAS trouvée");
        Class cl = friend.getClass();
        System.out.println("Classe de l'applet visualise (2): " + cl.getName());

        try { Thread.sleep(5); }catch (Exception e) {}
        AppletVisualisation x = (AppletVisualisation)friend;
        x.setNomPieceChoisie(np);
     }

     private javax.swing.JButton jButton2;
     private javax.swing.JComboBox CBListe;
     private javax.swing.JButton jButton1;
}
```

Il nous reste à placer les deux applets dans un fichier jar liste.jar. On peut alors constater que tout se passe bien comme prévu :


Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://localhost:8080//ShopAchats.html
Windows Hotmail Personnaliser les liens Windows Media Invasive species in B...
Search:
Bookmarks Toolbar Folder
Quick Searches
Google Quicksearch
Google Recherche avan...
Answers.com Dictionary ...
Stock Symbol Quicksearch
Wikipedia Quicksearch
Firefox and Mozilla Links
Firefox Start Page
Firefox Central
Themes and Extensions
Mozilla.com
Mozilla Developer Center
MozillaZine
Mozilla Store
Get Involved - Help spre...
MSN.com
RealPlayer Home Page
Links
RealPlayer
Guide des stations de radio
Accueil Google Earth
Google Earth : le guide d...
Media
Choisissez le livre qui vous intéresse :-)
Java II
Java II
La vie de Lady Di
Sexe et mensonges
Les orgies romaines
Kamasutra pour les nuls
Complot à l'Inpres
Les orgies romaines
Acheter
Terminer
Vous choisissez l'ouvrage :
Les orgies romaines

Bien sûr, la première applet, à des fins de démonstration, recherche d'autres renseignements qu'elle affiche dans la console :

```
Initialisation de l'applet ...
(Re)démarrage de l'applet ...
On a appuyé sur Entrer :-)
Initialisation de l'applet ...
(Re)démarrage de l'applet ...
On a appuyé sur Entrer :-)
Pièce cherchée : Les orgies romaines
Début de recherche
Applet trouvée : visualise
Applet trouvée : liste
Applet visualise trouvée
Classe de l'applet visualise (2): AppletVisualisation
Reception du nom choisi : Les orgies romaines
```

Il convient de bien remarquer cependant que **les deux applets doivent se trouver dans le même jar**. En effet,

> les classes de chaque jar sont chargées en mémoire par un objet chargeur de classe différent qui travaille dans une sandbox différente !

Or, **les méthodes getApplet() et getApplets() ne fonctionnent que pour un chargeur de classe commun**. Si donc on avait créé deux jars différents, la communication était alors impossible.

Remarque

S'il s'agissait de faire communiquer deux applets situées dans des pages différentes, il faudrait impérativement un relais du côté du serveur Web : ce serait le rôle d'une **servlet**, complément Java à un serveur Web et traitant ce genre de requêtes (le serveru Web ne sert que d'auxiliaire pour les transferts http). Nous en reparlerons dans les chapitres consacrés à la programmation Web (servlets et Java Server Pages) qui se trouvent dans Java III.

12. L'utilisation des packages

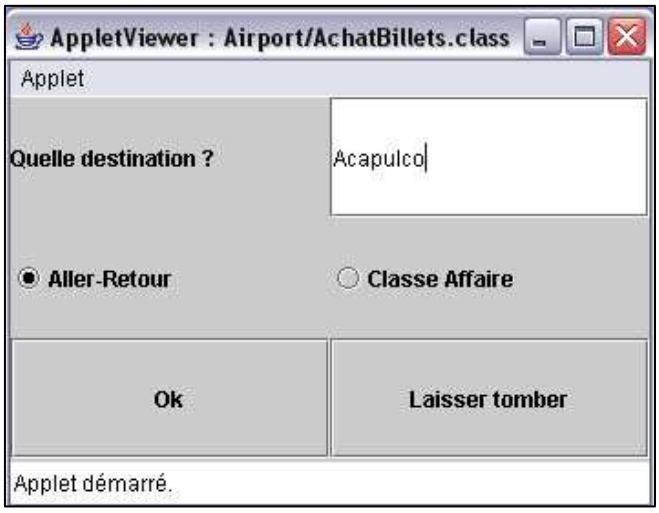
Supposons avoir développé une applet AchatBillets dans un package Airport :

AchatBillets.java

```
/*
 * AchatBillets.java
 *
*/

package Airport;

public class AchatBillets extends javax.swing.JApplet
{
   public AchatBillets()   { initComponents(); }
```

```
    private void initComponents() { … }

    private javax.swing.JButton jButton2;
    private javax.swing.JRadioButton jRadioButton1;
    private javax.swing.JLabel jLabel1;
    private javax.swing.JButton jButton1;
    private javax.swing.JRadioButton jRadioButton2;
    private javax.swing.JTextField jTextField1;
}
```

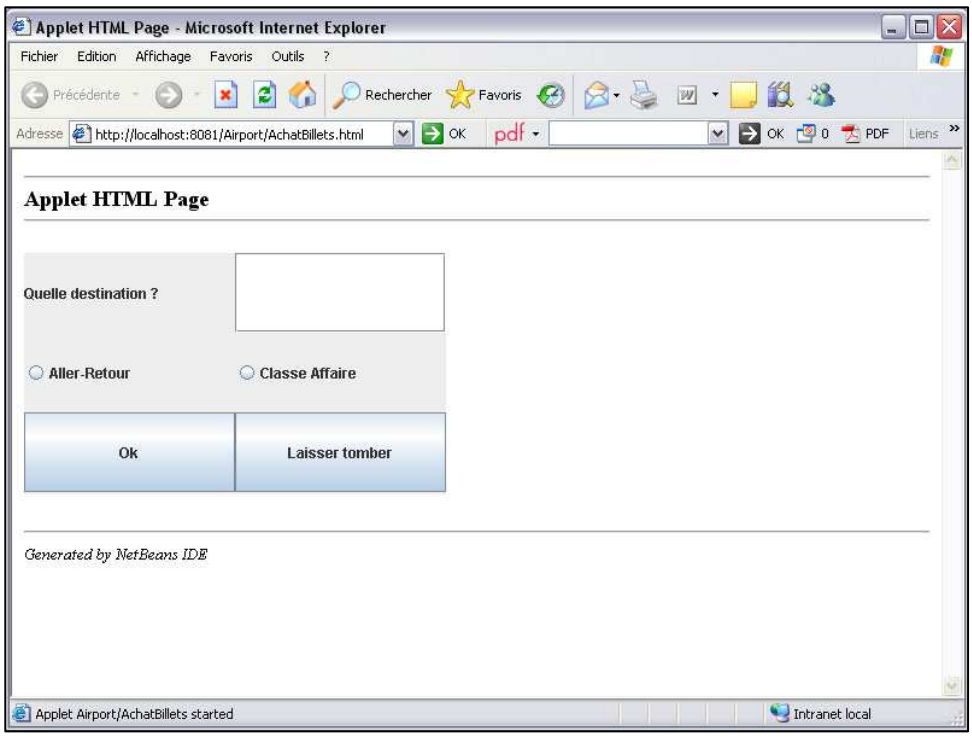
Après l'avoir testée dans le viewer d'applet :



on peut voir dans la page html générée comment le tag applet a été modifié :

```
<APPLET codebase=.. code="Airport/AchatBillets.class" width=350
        height=200></APPLET>
```

On remarquera l'attribut codebase pour permettre de situer le répertoire du package. L'exécution de cette page se passe sans problèmes :



Construisons à présent un jar avec notre applet :

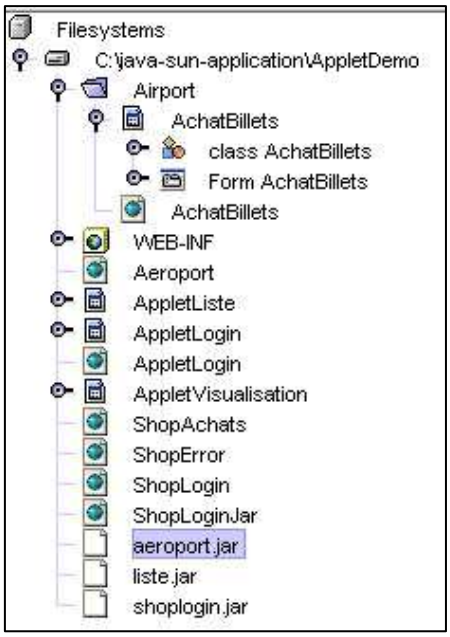
```
C:\java-sun-application\AppletDemo>jar cvf aeroport.jar Airport/*.class
manifest ajoutÚ
ajout : Airport/AchatBillets.class(entrÚe = 1441) (sortie = 805)(44% compressÚs)
```

et utilisons-le dans une page html :

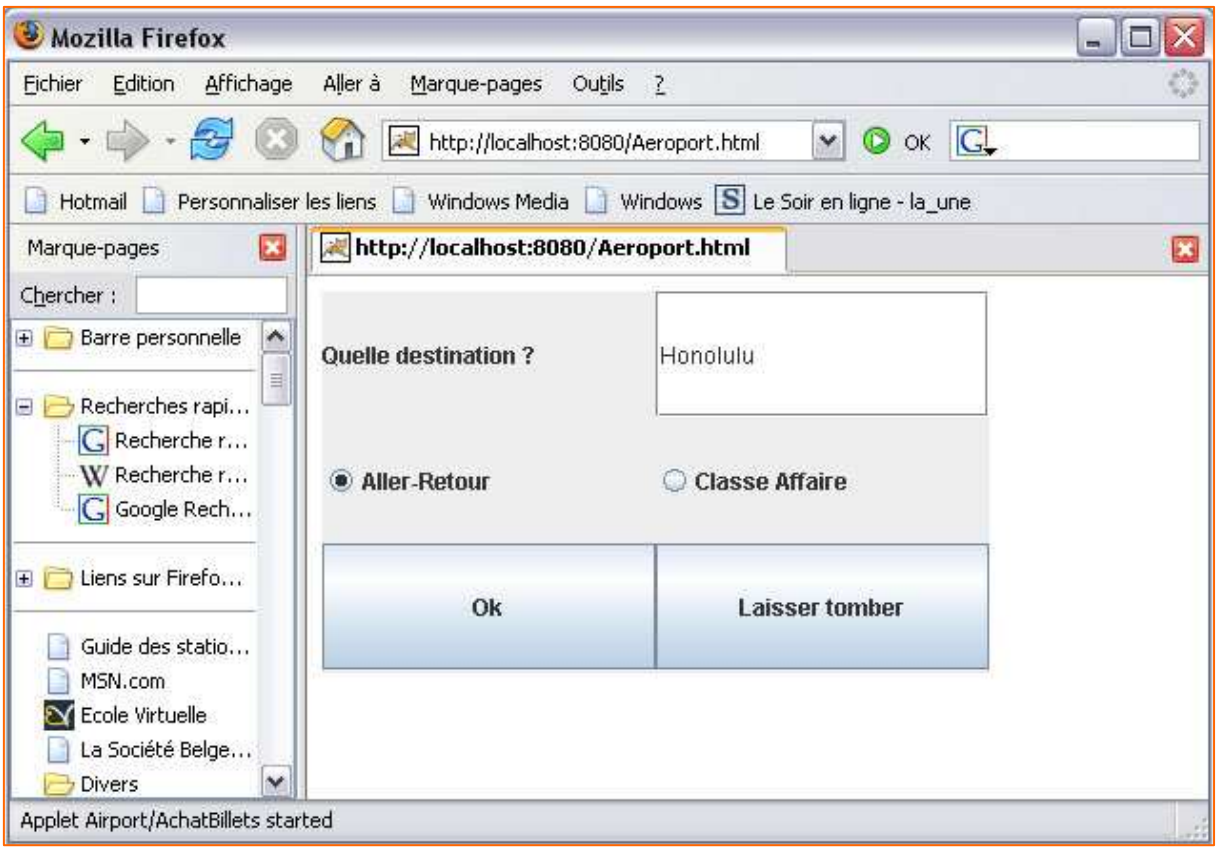
Aeroport.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<HTML>
 <HEAD>
  <TITLE></TITLE>
 </HEAD>
 <BODY>

  <APPLET code="Airport/AchatBillets.class" archive="aeroport.jar" width=350
        height=200></APPLET>

 </BODY>
</HTML>
```

➔

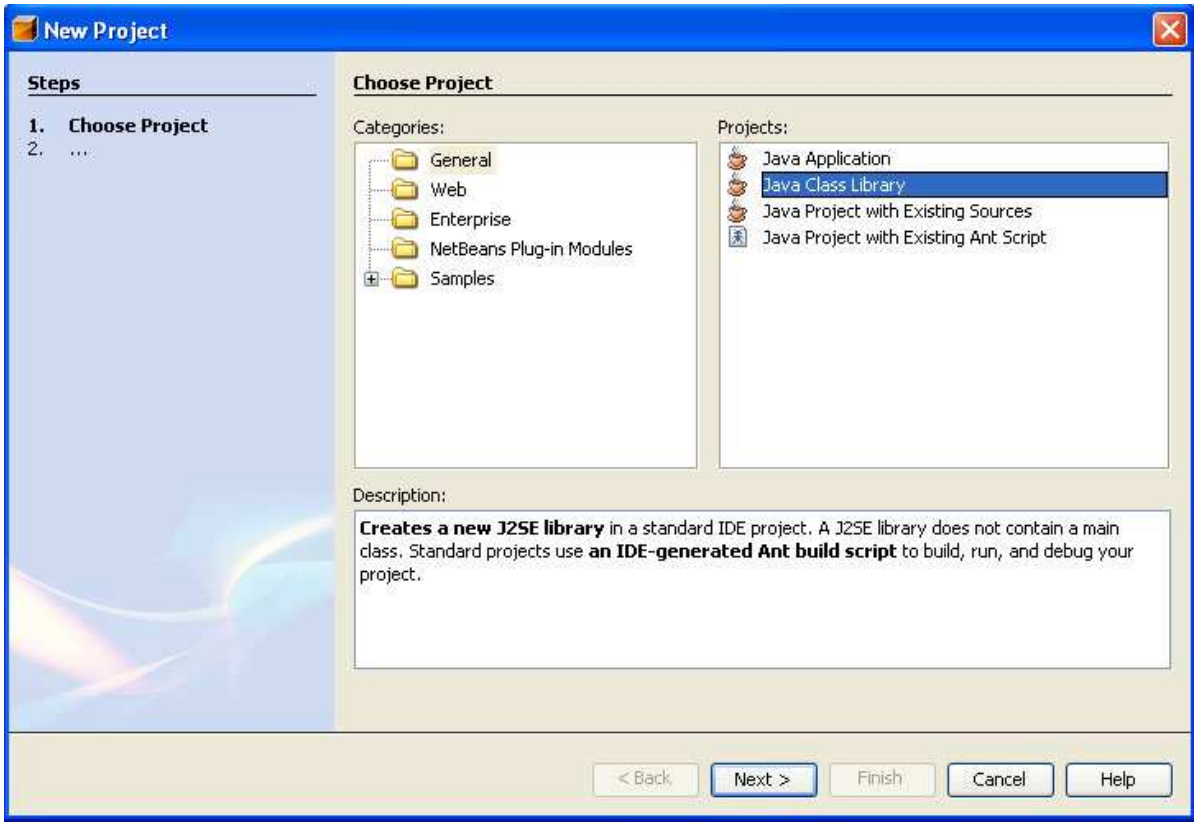


Une fois la page et le jar copié dans le répertoire ROOT de Tomcat, on obtient sans difficultés :



13. Le développement d'une applet dans NetBeans 5.*

Dans l'EDI NetBeans, on crée un projet destiné à forger une applet au moyen de :



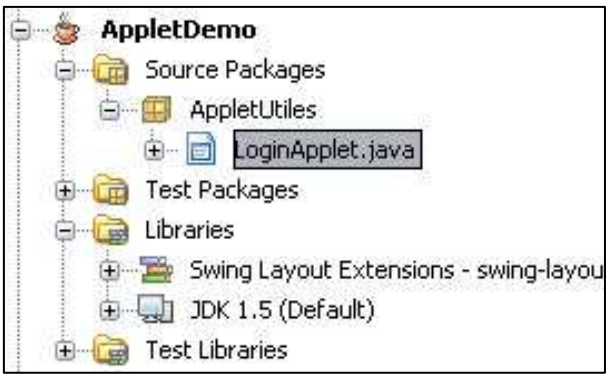
Comme d'habitude, il suffit alors de créer un package (disons "AppletsUtiles"), puis :

<clic droit> → New → JApplet …

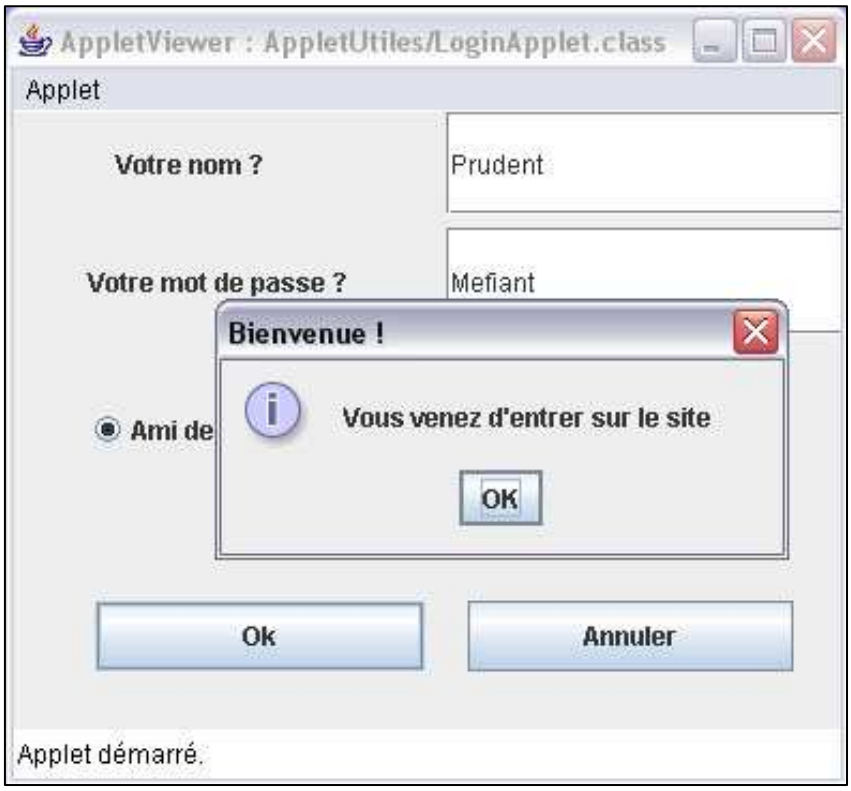
On peut donc rédiger le code de l'applet, par exemple :

LoginApplet.java

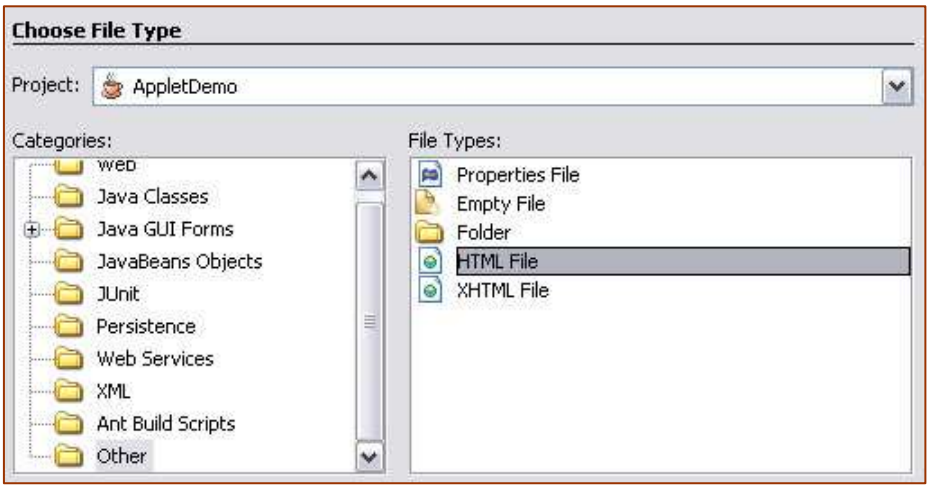
```
/*
 * LoginApplet.java
 *
*/

package AppletUtiles;

import javax.swing.*;

/**
 * @author  Vilvens
 */

public class LoginApplet extends javax.swing.JApplet
{
   public void init() { … }

   private void initComponents() { … }

   private void jButton1ActionPerformed(java.awt.event.ActionEvent evt)
   {
      System.out.println("On a appuyé sur Ok :-) ");
      JOptionPane.showMessageDialog(this, "Vous venez d'entrer sur le site", "Bienvenue !",
                  JOptionPane.INFORMATION_MESSAGE,null);
   }

   private javax.swing.ButtonGroup buttonGroup1;
   private javax.swing.JButton jButton1;
   private javax.swing.JButton jButton2;
   private javax.swing.JLabel jLabel1;
   private javax.swing.JLabel jLabel2;
   private javax.swing.JRadioButton jRadioButton1;
   private javax.swing.JRadioButton jRadioButton2;
   private javax.swing.JRadioButton jRadioButton3;
   private javax.swing.JTextField jTextField1;
   private javax.swing.JTextField jTextField2;
}
```



Le choix de "Run file" dans le menu contextuel obtenu par un clic droit sur le nom d'l'applet lance le viewer d'applet :



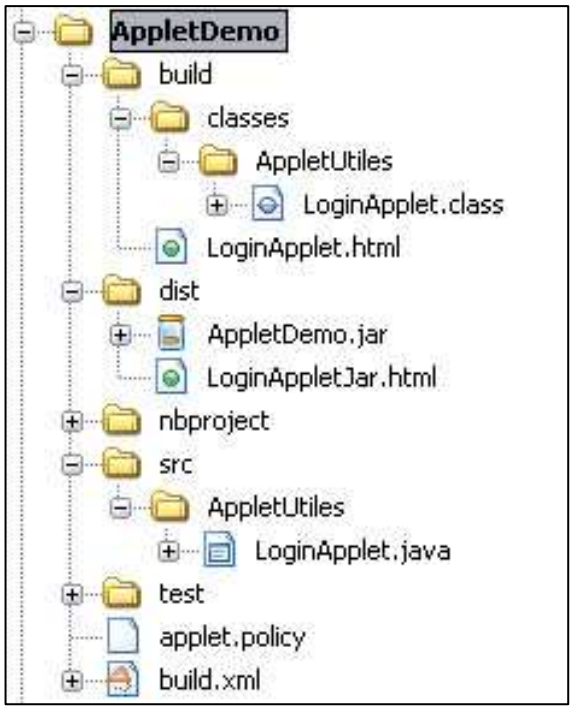
On peut à présent penser à créer une page HTML particulière pour héberger notre applet – bien sûr, nous la distribuerons sous forme de jar.



LoginAppletJar.html

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">

<html>
  <head>
    <title></title>
  </head>
  <body>
     <H3>Bienvenue à Sparte :-( </H3>
<P>
<APPLET code="AppletUtiles/LoginApplet.class" archive="AppletDemo.jar" width=450
height=300></APPLET>
</P>

  </body>
</html>
```

le jar en question étant obtenu en réalisant le "Build Project" du projet - celui-ci se trouve dans le répertoire dist, comme le montre l'onglet Files :



Cependant, l'exécution directe de la page html ou encore son appel depuis un browser, après l'avoir copiée en compagnie du jar dans le répertoire ROOT de Tomcat, se solde, dans les deux cas, par un échec ☹. Que se passe-t-il ?

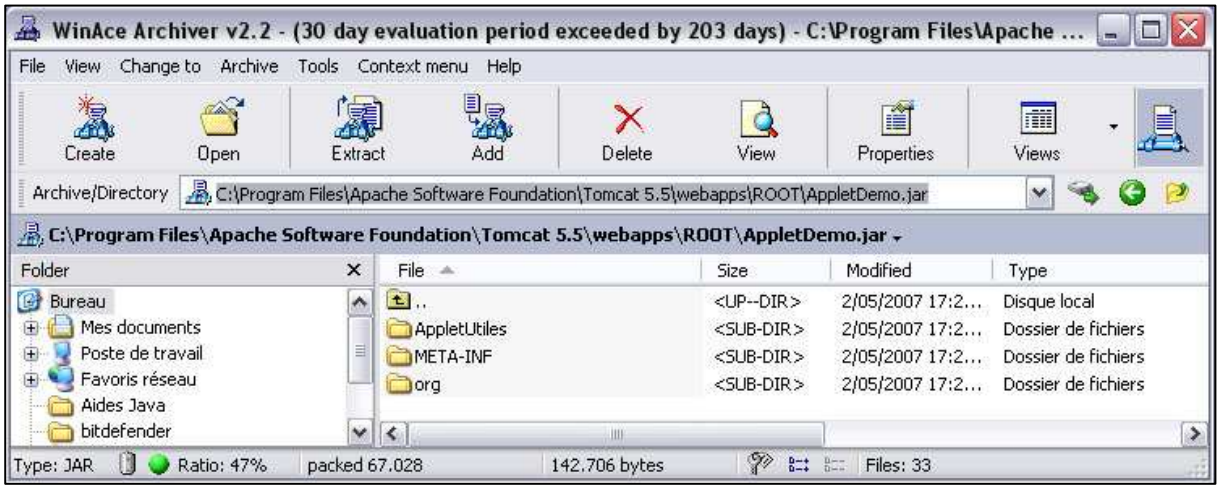
En fait, la console Java nous renseigne : il semblerait qu'un layout ne soit pas trouvé … Et, de fait, les sympathiques GUIs de Netbeans sont réalisés avec des layouts propriétaires qui se trouvent dans un jar nommé swing-layout-1.0.jar – il se trouve dans C:\Program Files\netbeans-5.5\platform6\modules\ext. Nous pourrions évidement penser à une solution faisant intervenir une clause Class-Path dans le manifeste du jar de notre applet.

Mais il y a mieux : nous allons ***configurer notre projet pour que la génération du jar de notre projet pense à inclure le jar complémentaire***. Pour ce faire, il suffit d'insérer dans le fichier build.xml (accessible par l'onglet Files) le tag suivant :

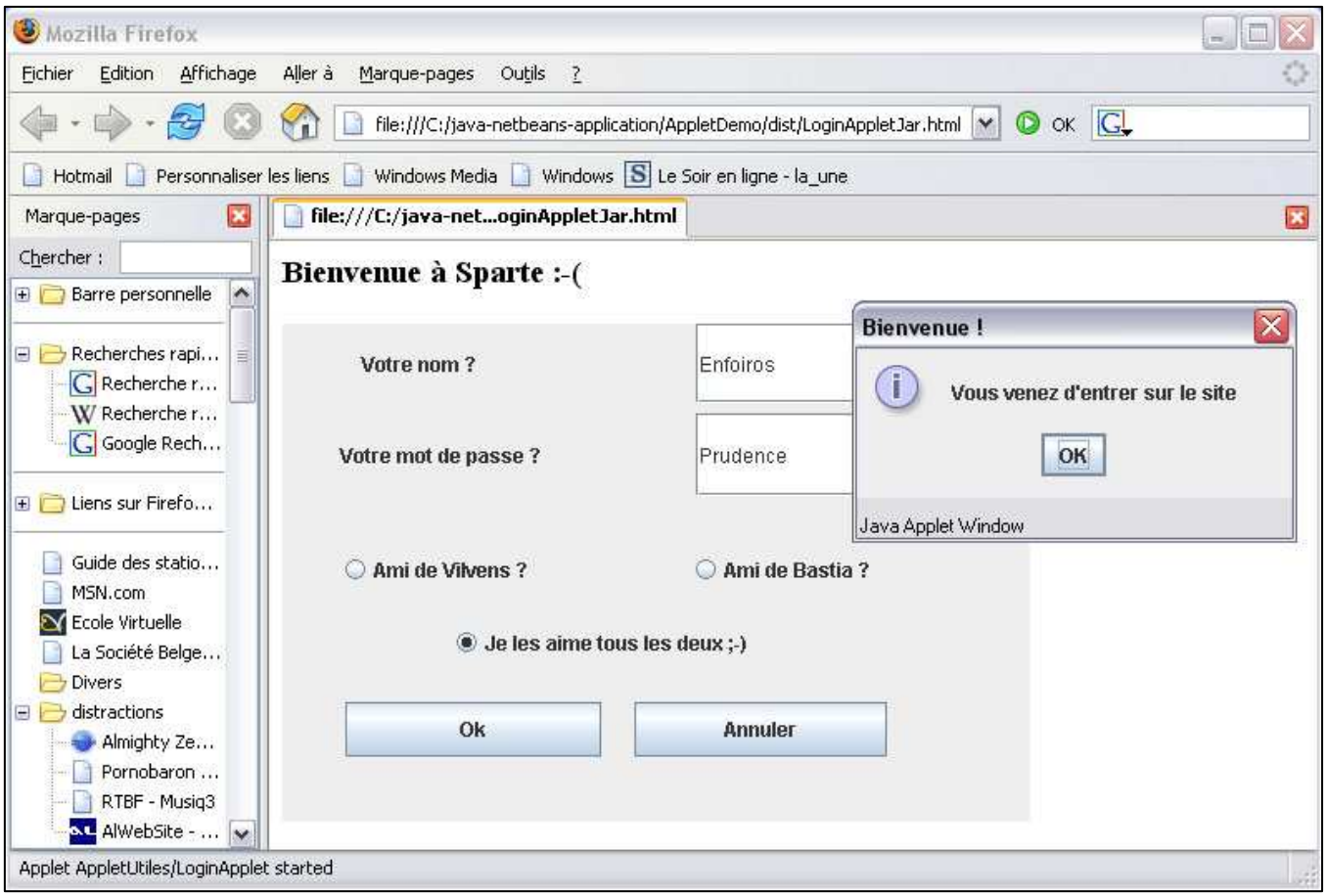
build.xml

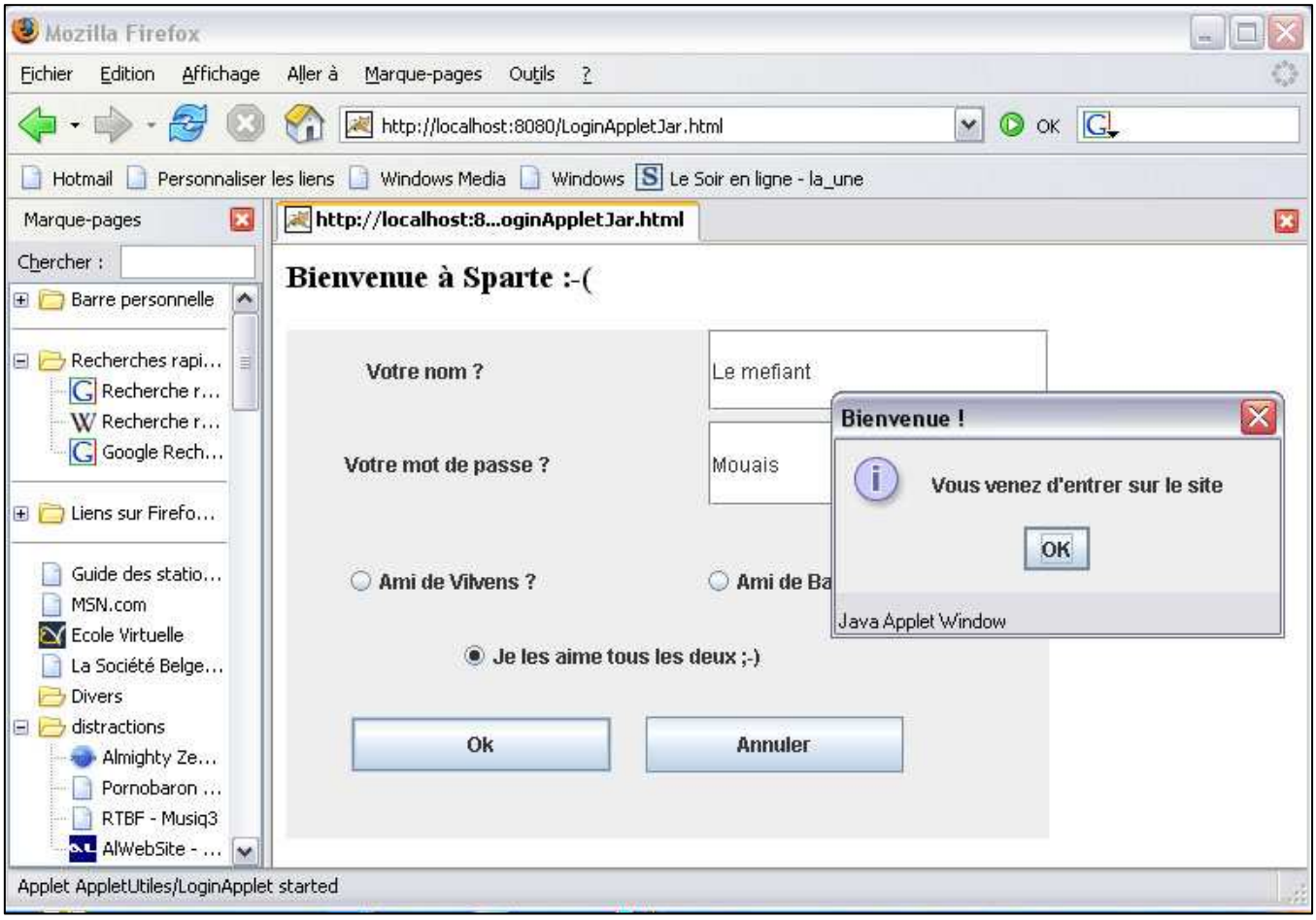
```
<?xml version="1.0" encoding="UTF-8"?>
<project name="AppletDemo" default="default" basedir=".">
   <description>Builds, tests, and runs the project AppletDemo.</description>
   <import file="nbproject/build-impl.xml"/>
   <target name="-post-jar">
      <jar update="true" destfile="${dist.jar}">
         <zipfileset src="${libs.swing-layout.classpath}"/>
      </jar>
   </target>
</project>
```

Une nouvelle génération du jar de notre projet va cette fois ajouter le jar des layouts :



On peut dès lors exécuter la page depuis Netbeans ou utiliser Tomcat : tout fonctionne ☺ !


Mozilla Firefox
Fichier Edition Affichage Aller à Marque-pages Outils ?
file:///C:/java-netbeans-application/AppletDemo/dist/LoginAppletJar.html
OK
Hotmail Personnaliser les liens Windows Media Windows Le Soir en ligne - la_une
Marque-pages
Chercher :
Barre personnelle
Recherches rapi...
Recherche r...
Recherche r...
Google Rech...
Liens sur Firefo...
Guide des statio...
MSN.com
Ecole Virtuelle
La Société Belge...
Divers
distractions
Almighty Ze...
Pornobaron ...
RTBF - Musiq3
AlWebSite - ...
file:///C:/java-net...oginAppletJar.html
Bienvenue à Sparte :-(
Votre nom ?
Enfoiros
Votre mot de passe ?
Prudence
Bienvenue !
Vous venez d'entrer sur le site
OK
Java Applet Window
Ami de Vilvens ?
Ami de Bastia ?
Je les aime tous les deux ;-)
Ok
Annuler
Applet AppletUtiles/LoginApplet started


Mozilla Firefox
Fichier Edition Affichage Aller à Marque-pages Outils ?
http://localhost:8080/LoginAppletJar.html
OK
Hotmail Personnaliser les liens Windows Media Windows Le Soir en ligne - la_une
Marque-pages
Chercher :
Barre personnelle
Recherches rapi...
Recherche r...
Recherche r...
Google Rech...
Liens sur Firefo...
Guide des statio...
MSN.com
Ecole Virtuelle
La Société Belge...
Divers
distractions
Almighty Ze...
Pornobaron ...
RTBF - Musiq3
AlWebSite - ...
http://localhost:8...oginAppletJar.html
Bienvenue à Sparte :-(
Votre nom ?
Le mefiant
Votre mot de passe ?
Mouais
Bienvenue !
Vous venez d'entrer sur le site
OK
Java Applet Window
Ami de Vilvens ?
Ami de Ba
Je les aime tous les deux ;-)
Ok
Annuler
Applet AppletUtiles/LoginApplet started