Applet

Un applet est un composant qui s'exécute dans le contexte d'un autre programme

Exemples :

- « Java Applet » - (navigateur internet),
- « Flash movie » - (navigateur internet),
- « Applet gnome » - (environnement graphique),
- ....

Applet JAVA

Le mécanisme d'Applet Java a été proposé avec la première version de Java,

L'environnement de développement de Sun propose un AppletViewer pour faciliter le développement.

Navigateur HotJava

En 1995, des développeurs Java propose un Navigateur WebRunner qui est un clone du navigateur Mosaic

Il a ensuite été appelé HotJava.

Il a été le premier navigateur à supporter les Applets. Il n'est plus supporté.

Navigateur Xbrowser

http://xbrowser.armondavanes.com/

Il y a eu également un browser open source

Applet JAVA

Le code est chargé depuis un serveur,

Il s'exécute à l'intérieur d'un navigateur ou d'un appletviewer

Il est lancé à travers une page html

Applet Sandbox

L'applet s'exécute dans le navigateur Internet dans un environnement sécurisé

Il y a des limitations sur l'utilisation de l'écran, du disque, du réseau, ...

Rappel IHM JAVA

Les composants graphiques sont dérivés des classes :

`Component` pour AWT (Abstract Window Toolkit);

`JComponent` pour Swing;

AWT est plus ancien et compact. Swing offre plus de fonctionnalités.

Page HTML

Pour avoir un « Applet » permettant d'afficher Bonjour définit dans une Classe Bonjour.

```
<HTML>
<HEAD>
<TITLE> Un applet simple </TITLE>
<HEAD>
<APPLET CODE="Bonjour.class" WIDTH="150" HEIGHT="25">
</APPLET>
</BODY>
</HTML>
```

Balises APPLET

- `WIDTH` : largeur,
- `HEIGHT` : hauteur,
- `CODE` : classe correspondant à l'Applet,
- `CODEBASE` : chemin des classes (url),
- `ARCHIVE` : permet de préciser une ou des archive(s) JAR.

paramètres

On peut dans la page HTML définir un ensemble de paramètres sous la forme de couples (nom, valeur).

```
<APPLET CODE="Bonjour.class"
  WIDTH="150" HEIGHT="50">
<PARAM NAME="nom" VALUE="toto">
</APPLET>
```

Permet de définir une variable nom associée à la valeur toto.

OBJECT

```
<APPLET codebase = "a/b/c ...>
<PARAM name="codebase" value="my.jar">
</APPLET>
```

Donne

```
<OBJECT ...>
<PARAM name="java_codebase" value="a/b/c">
<PARAM name="codebase" value="my.jar">
</OBJECT>
```

Balises OBJECT

- java_code,
- java_codebase,
- java_archive,
- java_object,
- java_type.

Balise EMBED

```
<EMBED type="application/x-java-applet;jpi-version=1.4.1"
   width="200"
    height="200" align="baseline" code="XYZApp.class"
    codebase="html/" model="models/HyaluronicAcid.xyz"

   pluginspage="http://java.sun.com/j2se/1.4.1/download.html">
<NOEMBED>
    No Java 2 SDK, Standard Edition v 1.4.1 support for
   APPLET!!
</NOEMBED>
</EMBED>
```

Applet Bonjour

```
import java.applet.Applet;
import java.awt.Graphics;


public class Bonjour extends
  Applet {
 public void paint{Graphics g) {
   g.drawString("Bonjour", 50, 25);
}
```

paint

Les méthodes `paint` et `repaint` permettent l'affichage du composant.

```
public void paint{Graphics g) {}
public void repaint()
```

Classe Applet

```
java.lang.Object
 java.awt.Component
  java.awt.Container
    java.awt.Panel
     java.applet.Applet
```

Cycle de vie

Au départ on appel la méthode `init()`.

Elle permet de faire les initialisations.

```
public void init ()
{
 System.out.println("init");
}
```

Cycle de vie

La méthode `start()` correspond au début du traitement de l'Applet.

```
public void start ()
{
 System.out.println("start");
}
```

Cycle de vie

La méthode `stop()` permet de stopper l'exécution.

```
public void stop()
{
 System.out.prinln("stop");
}
```

Cycle de vie

La méthode `destroy()` prépare le déchargement de l'applet.

```
public void destroy()
{
 System.out.prinln("destroy");
}
```

getParameter

La méthode `getParameter()` permet d'obtenir la valeur associée à un paramètre.

```
String nom =
  this.getParameter("nom");
```

showStatus

La méthode `showStatus()` permet d'afficher un texte dans la barre d'état.

```
this.showStatus("l'Applet marche");
```

Taille

On peut obtenir la taille de l'Applet avec les méthodes `getSize().width` et `getSize().height`.

```
// Pour dessiner un rectangle de la taille de
  l'Applet jdk 1.1
g.drawRect(0,0,size().width-1, size().height-
  1);
// ensuite
g.drawRect(0,0,getSize().width-1,
  getSize().height-1);
```

Evénements

L'interface graphique interagit avec l'utilisateur à travers des événements.

Cela permet d'économiser les ressources.

En effet on a des actions à faire que si il y a un événement.

Souris

Par exemple on peut obtenir la position de la sourie de la façon suivante :

```
import java.awt.Event;...
// jdk 1.1
public boolean mouseDown(Event
  event, int x, int y){
System.out.println("Click x "+x+" y
  "+y);
return true; }
```

Evénements

Pour simplifier la gestion des événements, il existe des comportements standard.

Pour pouvoir y associer des fonctions d'un programme on va utiliser un système de notification.

On va enregistrer des auditeurs (Listener) pour être informé lorsqu'il y a un événement.

ActionListener

Création d'un BoutonListener.

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
public class BoutonListener implements
  ActionListener {
public void actionPerformed(ActionEvent
  e)
{System.out.println("Click Bouton ");}
}
```

On va reprendre l'exemple de la position de la souris.

```
import java.awt.event.MouseListener;
import java.awt.event.MouseEvent;


public class Souris implements MouseListener {
    public void mousePressed(MouseEvent e) {..}
    public void mouseReleased(MouseEvent e) {..}
    public void mouseEntered(MouseEvent e) {..}
    public void mouseExited(MouseEvent e) {..}
    public void mouseClicked(MouseEvent e)
     { System.out.println("Click x "+e.getX()+" y
  "+e.getY()); }
}
```

addMouseListener

Il faut ensuite enregistrer l'auditeur (Listener).

```
public void init()
     {...
      this.addMouseListener(new
       Souris());
      ...
     }
```

Bouton

Nous allons utiliser le même principe pour un bouton.

Le fait d'activer le bouton est appelé « actionPerformed ».

Nous allons associer un affichage avec un auditeur (Listener) au Bouton.

ActionListener

Création d'un BoutonListener.

```
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
public class BoutonListener implements
  ActionListener {
public void actionPerformed(ActionEvent e)
 {
  System.out.println("Click Bouton ");
 }
}
```

addActionListener

Création d'un Bouton et association d'une action.

```
public void init ()
{
 ...
 Button bouton= new Button("Click");
 add(bouton);
 bouton.addActionListener(new
 BoutonListener());
 ...
}
```

Exemple Applet Horloge

Nous allons réaliser un Thread permettant d'afficher l'heure dans « un Applet ».

Pour cela nous allons créer un Thread au démarage de l'Applet.

Nous allons l'arréter à la fin.

Il va afficher l'heure toutes les secondes.

start Horloge

```
public void start() {
if (horlogeThread == null) {
 horlogeThread = new Thread(this,
  "Horloge");
 horlogeThread.start()
}
```

run Horloge

```
public void run() {
 Thread monThread = Thread.currentThread();
 while (horlogeThread == monThread) {
  repaint()
  try {Thread.sleep(1000);}
catch (InterruptedException e) {
   // Si interruption
    }
  }
}
```

paint Horloge

```
public void paint(Graphics g) {
 Calendar cal = Calendar.getInstance();
 Date date = cal.getTime();
 //format d'affichage
 DateFormat dateFormateur =
  DateFormat.getTimeInstance();
 //affichage
 g.drawString(dateFormateur.format(date), 5,
  10);
}
```

stop Horloge

```
public void stop() {
horlogeThread = null;
}
```

Java Web Start

Une évolution possible des applets est java web start.

Ce mécanisme permet de lancer une application java à partir d'un navigateur Internet.

L'application ne s'exécute pas dans le browser il y a moins de limitation

Fin