Java - Introduction à Swing

Code: java-swing

A COMPLETER !

Originaux

url: **http://tecfa.unige.ch/guides/tie/html/java-swing/java-swing.html**

url: **http://tecfa.unige.ch/guides/tie/pdf/files/java-swing.pdf**

Auteurs et version

- Daniel K. Schneider

- Version: 0.4 (modifié le 11/4/01)

Prérequis

Module technique précédent: java-intro

Module technique précédent: java-awt

Module technique précédent: java-xml et java-sql (pour les contenus)

Modules

Module technique suivant: ....

Objectifs

- Accompagement des tutorials Swing des Sun
- Montrer des petites applications faites avec Swing
- A faire: Exemples pour éditer du XML

1. Table des matières

1. Table des matières 3
2. Introduction à JFC/SWING 4
2.1 Features 4
2.2 Applets avec Swing 5
2.3 Sites / Tutoriels Swing 5
2.4 Un exemple complet d'une simple application 6
3. Le concept modèle - vue - contrôleur 10
A.Le modèle 11
B.La vue et le contrôleur 11
C.Et Swing ? 11
4. Simples JTables 12
4.1 Une JTable pour visualiser une requête SQL 12
A.Le Layout de l'UI 12
B.User Interaction 14
C.Le modèle 15
4.2 Une JTable pour visualiser un fichier XML 16
5. Examples à adapter à vos besoins 17
5.1 XML / Jtrees 17

2. Introduction à JFC/SWING

- JFC / Swing est le nouveau paradigme GUI de Java
  - marche avec les JDK 1.2 / 1.3 (donc toutes les applications Java 2)
  - avec Netscape 4.x / IE 5.5 ET le plugin Java 1.2. de Sun (applets)
  - partiellement avec le JDK 1.1 plus les classes Swing (JFC 1.1)
- Package: javax.swing

2.1 Features

- un meilleur support pour créer des GUI plus sophistiqués
  - Pour voir ce que cela donne, affichez: **http://tecfa.unige.ch/guides/java/tutorial/uiswing/components/components.html**
- Pluggable Look & Feel Support
  - un même programme peut utiliser Java et / ou Windows look & feel
- Java 2D API (JDK 1.2)
  - Graphisme, texte et images de haute qualité
- Drag & Drop support (JDK 1.2)

2.2 Applets avec Swing

Il existe 2 solutions pour faire des applets Swing qui marchent avec un browser standard

1. Utiliser les classes Swing 1.1 avec Java 1.1x
   - dans ce cas il faut soit copier l'archive swing.jar dans le répertoire java de votre browser, soit inclure l'archive dans l'applet (solution lourde en temps de chargement)
   - Attention: Swing 1x ne contient pas tout (Java 2D, Java3D, etc.)
2. Utiliser le Java Plugin de Sun qui permet de faire tourner la Java 2 plateforme (Java 1.2 / 1.3) qui contient Swing 2.

   url: **http://java.sun.com/products/plugin/** (Java plug-in page)

   url: **http://java.sun.com/products/plugin/1.3/docs/tags.html** (HTML specification)

2.3 Sites / Tutoriels Swing

- Voir la toolbox pour le tutoriels
- Chez / de Sun:

  url: **http://java.sun.com/products/jfc/** (fouillez dans ce JFC portail)

  url: **http://tecfa.unige.ch/guides/java/tutorial/uiswing/index.html** (copie locale)

2.4 Un exemple complet d'une simple application

Exemple 2-1: Une simple SwingApplication (Sun Java Tutorial)

- Voir **http://tecfa.unige.ch/guides/java/staf2x/ex/swing/simple/SwingApplication.java**
- Tutoriel: **http://tecfa.unige.ch/guides/java/tutorial/uiswing/start/swingTour.html**


```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class SwingApplication {

    private static String labelPrefix = "Number of button clicks: ";
    private int numClicks = 0;
```

```
public Component createComponents() {
 // Define a label widget
 final JLabel label = new JLabel(labelPrefix + "0    ");

 // Define a press button widget
 JButton button = new JButton("I'm a Swing button!");
 // ALT-i will trigger it also
 button.setMnemonic('i');
 // add an actionPerformed method to the buttons Action Listener.
 button.addActionListener(new ActionListener() {
      // each time the user clicks, the text of the Label gets changed
      public void actionPerformed(ActionEvent e) {
          numClicks++;
          label.setText(labelPrefix + numClicks);
      }
     });
    label.setLabelFor(button);

 // We need a container (panel) to put the label and the button

 JPanel pane = new JPanel();
 // A Border Factory will create an Empty Border we will use
 pane.setBorder(BorderFactory.createEmptyBorder(30, 30, 10, 30));
 // We use a simple grid layout
 pane.setLayout(new GridLayout(0, 1));
 pane.add(button);
 pane.add(label);
 return pane;
}
```

```
public static void main(String[] args) {

     //Create the top-level container and add contents to it.
     JFrame frame = new JFrame("SwingApplication");
     SwingApplication app = new SwingApplication();

     // The createComponents method will produce the pane with the 2 widgets
     // We then can add it to the contents of our frame
     Component contents = app.createComponents();
     frame.getContentPane().add(contents, BorderLayout.CENTER);

     //Finish setting up the frame, and show it.
     frame.addWindowListener(new WindowAdapter() {
          public void windowClosing(WindowEvent e) {
              System.exit(0);
          }
         });
     frame.pack();
     frame.setVisible(true);
    }
}
```

A retenir:

- Définir un "top-level" container

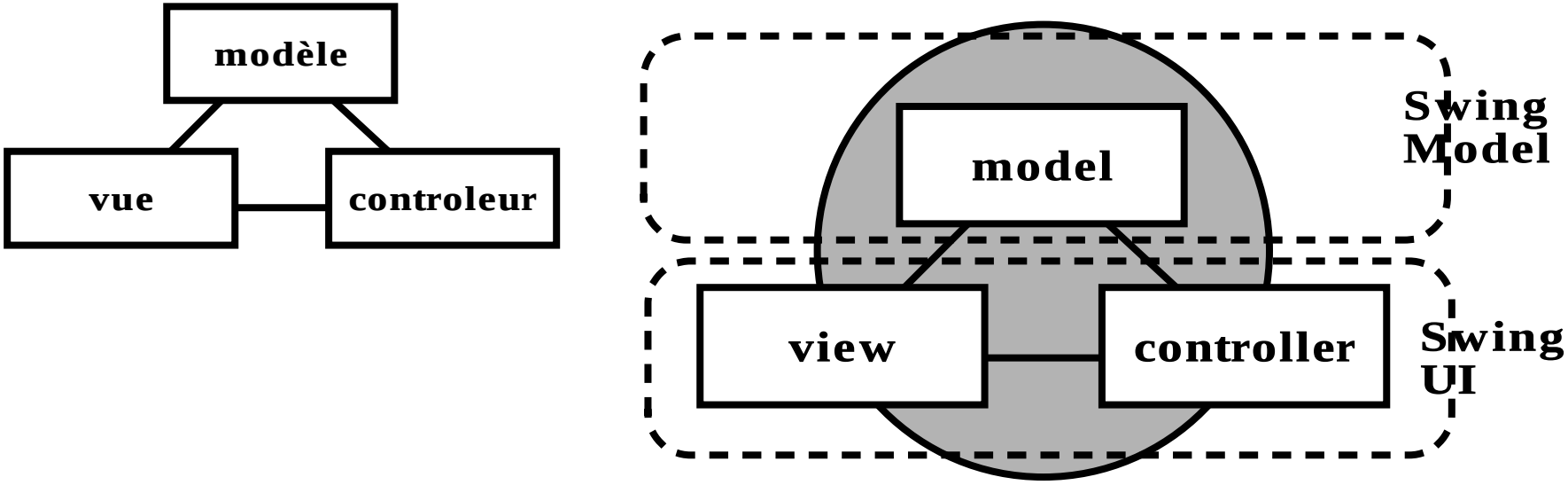
```
public class SwingApplication {
                    ...
       public static void main(String[] args) {
                         ...
      JFrame frame = new JFrame("SwingApplication");
      //..create the components to go into the frame...
      //...stick them in a container (the content pane of the frame)
     frame.getContentPane().add(contents, BorderLayout.CENTER);

      // Finish setting up the frame (make sure people can close it), and show it.
     frame.addWindowListener(new WindowAdapter() {
             public void windowClosing(WindowEvent e) {
                   System.exit(0);
                          }
                   });
             frame.pack();
             frame.setVisible(true);
       }       }
```

- Boutons et labels:

```
     final JLabel label = new JLabel(" ......   ");
     JButton button = new JButton("I'm a Swing button!");
     button.setMnemonic('i');
```

3. Le concept modèle - vue - contrôleur



Le modèle:

1. Le modèle représente les données de l'application ou autrement dit: l'état. Il ne connaît rien de ses contrôleurs ou de ses vues.

L'interface utilisateur:

2. La vue correspond à la représentation visuelle des données (dans leur état actuel!)
3. Le contrôleur gère l'interaction utilisateur avec le modèle. Il intercepte le user input (dans la vue) et le traduit en changes dans le modèle.

A. Le modèle

possède (en règle générale) 4 types de méthodes

1. Interrogation de son état interne (lire)
2. Manipulation de son état interne (changer, détruire)
3. Ajouter et enlever des évent listeners
4. exécuter (fire) des événements

B. La vue et le contrôleur

possède ces 3 types de méthodes:

5. peinture
6. retourne des informations géométriques
7. gestion d'événements UI (AWT comme click, ...)

C. Et Swing ?

- supporte ce type de modélisation explicitement avec ses widgets JList, JTable, JTree, JEditorPane, etc.
- En gros: on peut programmer l'UI pour qu'il observe l'état des données et fasse la mise en page et inversément [... à expliquer mieux]

4. Simples JTables

4.1 Une JTable pour visualiser une requête SQL

Exemple 4-1: Update / Query MySQL Swing Applet for COFFEES

url: **http://tecfa.unige.ch/guides/java/staf2x/ex/jdbc/coffee-break/query-applet/**

- clicker sur MySqlUpdateSwingApplet.html
- (voir aussi module TIE: java-mysql pour JDBC et d'autres exemples)

A. Le Layout de l'UI

- En gros on met des boites dans des boites, en commençant par les petits
- c'est plus simple que certains layouts compliqués

```
public void init () {

 // The Container
 container = getContentPane();

 // The update panel

 JPanel updatePanel = new JPanel ();
 updateTextField = new JTextField
     ("UPDATE COFFEES SET SALES = 75 WHERE COF_NAME LIKE 'Espresso'",45);
        updatePanel.add(updateTextField, BorderLayout.NORTH);
```

```
// The query panel (label + select field)
queryPanel = new JPanel ();
queryTextField = new JTextField("SELECT * FROM COFFEES", 45);
       queryPanel.add(queryTextField, BorderLayout.NORTH );

// In the heart a JTable with and instance of our tableModel
tableModel = new QueryTableModel ();
JTable table=new JTable (tableModel);
table.setPreferredScrollableViewportSize(new Dimension(500, 200));
JScrollPane queryScrollPane = new JScrollPane(table);
queryPanel.add(queryScrollPane, BorderLayout.CENTER);

// The tabbed Pane and the whole container
JTabbedPane tabbedPane = new JTabbedPane ();
tabbedPane.addTab("Update", null, updatePanel, "Simple Update Window");
tabbedPane.addTab("Query", null, queryPanel, "Simple Query Window");
container.add(tabbedPane, BorderLayout.CENTER);

// Text in the bottom panel
       textArea = new JTextArea("HELLO",4,45);
       textArea.setEditable(false);
JScrollPane textScrollPane = new JScrollPane(textArea);
       JPanel bottomPanel = new JPanel();
       bottomPanel.add(textScrollPane,BorderLayout.CENTER);
container.add(bottomPanel, BorderLayout.SOUTH);
```

B. User Interaction

Action Listener

- On dit à chaque input widget (queryTextField et updateTextField) de s'enregistrer avec MySqlUpdateSwingApplet (la classe applet qui implémente ActionListener)

```
//Add a Listener for the queryTextField
queryTextField.addActionListener(this);
updateTextField.addActionListener(this);
```

Gestion des événements

- faut faire qc quand l'utilisateur fait qc dans un input field

```
public void actionPerformed(ActionEvent evt) {
String text;
 Object TheWidget = evt.getSource();
 // user hit return in query Text Field
 if (TheWidget == queryTextField) {
     text = queryTextField.getText();
     //activate the setQuery method of the tableModel
     tableModel.setQuery (text);
     queryTextField.selectAll();
 }
 // must have hit return in the other field
 else if (TheWidget == updateTextField) {
     text = updateTextField.getText();
     update (text);
     updateTextField.selectAll();
```

C. Le modèle

- étend AbstractTableModel

Il contient plusieurs éléments:

- une méthode setQuery qui fait le grand travail:
  - requête à une base de données,
  - traduction des meta-data en un array pour les labels
  - traduction du ResultSet en une table flexible
    (vecteur cachedData qui contient des arrays rawLineData)
- des méthodes obligatoires pour le AbstractTableModel
  - ces méthodes vont chercher de l'information dans notre table

```
public int getColumnCount () {
        return noCols; }
public String getColumnName (int i) {
        return resColNames[i]; }
public int getRowCount () {
        return nRows; ; }
public Object getValueAt (int row, int col) {
        return ((String []) cachedData.elementAt(row))[col];
```

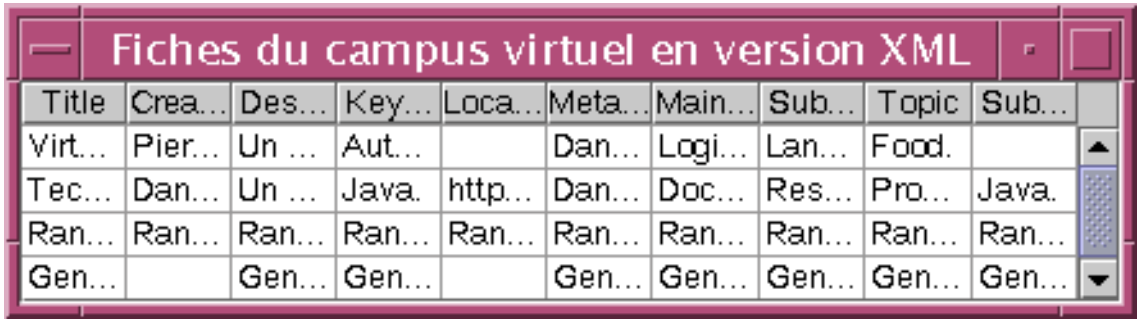
4.2 Une JTable pour visualiser un fichier XML

Exemple 4-2: Application Swing/JTable: Fiches du campus virtuel en version XML

url: **http://tecfa.unige.ch/guides/java/staf2x/ex/xml/fiches/v2/**

Attention: il faire tourner cela en local:

Installer Java 1.2 (pas testé avec 1.1 + Swing)
Mettre xerces.jar dans votre classpath
Soit: Copier les fichiers FicheModel2.* + FicheFrame2.* + FichePanel2.* + Fiches2.xml
(à Tecfa) monter le drive /comm/ sur PC et y aller
Taper 'java FicheFrame2'



- Note: pour transformer ca un applet il faut probablement apprendre à signer
- Explications:

url: **http://tecfa.unige.ch/guides/java/staf2x/ex/xml/fiches/**

5. Examples à adapter à vos besoins

5.1 XML / Jtrees

Exemple 5-1: Visualiser un fichier XML avec Swing/JTree (Version Xerces)

url: **http://tecfa.unige.ch/guides/java/staf2x/ex/xml/jtree**

L'original est distribué avec Xerces (**http://xml.apache.org/xerces-j/index.html**)

Environnement:

il faut faire tourner ca en local avec Java 1.2 (copier fichiers ou monter le drive)

xerces.jar doit être dans votre classpath (voir aussi les scripts)

Exécution "à la main":

(1) Vérifier le classpath vérifier que le sousrépertoire ui existe!

(2) Depuis le répertoire jtree(!):
Taper '**`java ui.TreeViewer ../formcont/formcont.xml`**'.
Vous pouvez visualiser n'importe quel fichier XML même via HTTP, par exemple:
'**`treeview http://tecfa.unige.ch/staf/staf-e/staf18/proj/proj11.xml`**'

Exécution avec un script (Unix ou Dos)

(1) aller dans le répertoire (pas besoin de mettre un classpath)

(2) taper 'treeview ../formcont/formcont.xml' (ou un autre fichier)

Exemple 5-2: Visualiser un fichier XML avec Swing/JTree (Version Xerces)

url: Copie locale: **http://tecfa.unige.ch/guides/java/staf2x/ex/xml/jtree-pfeiffer/**

- Ralf I. Pfeiffer, "XML Tutorials for programmers, Tutorial 3: Parsing XML Using Java", IBM

  url: **http://www-4.ibm.com/software/developer/education/tutorial-prog/parsing.html**
- Il s'agit ici d'une version antérieure par rapport à l'exemple 5-1 "Visualiser un fichier XML avec Swing / JTree (Version Xerces)" [16]
- J'ai adapté le code pour Xerces (modifications mineures)

Exécution "à la main":

(1) mettre Xerces dans le classpath

(2) aller dans le répertoire et taper 'java xxx fichier.xml' (xxx = une variante)

Réutilisation:

(1) Prendre une des versions (par exemple SimpleTreeView4.java)

(2) Recompiler ou encore prendre tous les fichiers *.class qui vont avec

(3) Prendre le fichier ui / DomTree.class et le remettre dans un sous-répertoire ui (ou dans un *.jar si vous voulez); "ui" est un package, ses classes doivent donc obligatoirement résider dans un sourépertoire du même nom !

(4) Adapter l'interface JTree comme dans SimpleTreeView4

Exemple 5-3: Les Jtree de IBM comme applet

url: **http://tecfa.unige.ch/guides/java/staf2x/ex/xml/jtree-applet/**

- faire un applet avec Xerces (et outils similaires) est lourd (1 MB de download)
- il faudrait plutôt installer des classes localement ou encore faire des applets avec des parseurs SAX "light-weight".