

## Les bases graphiques : la librairie « Standard Widget Toolkit » (suite)

## Table des matières

<b>LES BASES GRAPHIQUES : LA LIBRAIRIE « STANDARD WIDGET TOOLKIT » (SUITE) .....</b>	<b>1</b>
TABLE DES MATIERES .....	2
LES DIALOGS SWT (1).....	5
<i>les objectifs</i> .....	5
<i>les principes généraux</i> .....	5
<i>utilisation générique</i> :.....	5
LES DIALOGS SWT (2).....	6
<i>la boîte de message (la classe MessageBox)</i> .....	6
<i>deux exemples simples</i> .....	6
LES DIALOGS SWT (3).....	7
<i>la sélection des couleurs (la classe ColorDialog)</i> .....	7
<i>un exemple simple</i> .....	7
LES DIALOGS SWT (4).....	8
<i>la sélection des fichiers pour l'ouverture-fermeture (la classe FileDialog)</i> .....	8
<i>un exemple simple</i> .....	8
COMPLEMENTS SUR LES DIALOGS SWT.....	9
<i>la sélection des répertoires (la classe DirectoryDialog)</i> .....	9
<i>la création de ses propres dialogues</i> .....	9
LES LISTS.....	10
<i>les caractéristiques de la classe List</i> .....	10
<i>quelques méthodes</i> .....	10
<i>un exemple simple</i> .....	10
LES COMBOS.....	11
<i>les caractéristiques de la classe Combo</i> .....	11
<i>quelques méthodes</i> .....	11
<i>un exemple simple</i> .....	11
LES MENUS (1).....	12
<i>les caractéristiques de la classe Menu</i> .....	12
<i>quelques méthodes</i> .....	12
LES MENUS (2).....	13
<i>Un exemple de Menu (associé à la barre de menu)</i> .....	13
LES MENUS (3).....	14
<i>un exemple de menus pop-up</i> .....	14
LES DESSINS (1).....	15
<i>la classe GC (Graphical Context)</i> .....	15

## développement de plugins Eclipse

<i>quelques méthodes de la classe GC</i> .....	15
<i>le composite Canvas</i> .....	15
LES DESSINS (2) .....	16
<i>un exemple simple</i> .....	16
COMPLEMENTS : QUELQUES AUTRES WIDGETS.....	17
<i>le Slider</i> .....	17
<i>la Scrollbar</i> .....	17
<i>la Progressbar</i> .....	17
LES TABLES (1).....	18
<i>les tables (la classe Table)</i> .....	18
<i>quelques méthodes de la classe Table</i> .....	18
LES TABLES (2).....	19
<i>la classe TableColumn</i> .....	19
<i>quelques méthodes de TableColumn</i> .....	19
<i>la classe TableItem</i> .....	19
<i>quelques méthodes de TableItem</i> .....	19
LES TABLES (3).....	20
<i>un exemple simple</i> .....	20
LES TREES (1).....	21
<i>les arbres (la classe Tree)</i> .....	21
<i>quelques méthodes de la classe Tree</i> .....	21
LES TREES (2).....	22
<i>les nœuds d'un arbre (la classe TreeItem)</i> .....	22
<i>quelques méthodes de la classe TreeItem</i> .....	22
LES TREES (3).....	23
<i>un exemple simple</i> .....	23
LES DOSSIERS TABULÉS (1).....	24
<i>quelques généralités</i> .....	24
<i>quelques méthodes de TabFolder</i> .....	24
<i>quelques méthodes de TabItem</i> .....	24
LES DOSSIERS TABULÉS (2).....	25
<i>un exemple simple</i> .....	25
QUELQUES COMPOSITES SUPPLEMENTAIRES (1).....	26
<i>la classe SashForm</i> .....	26
<i>quelques méthodes de SashForm</i> .....	26
<i>un exemple simple</i> .....	26
QUELQUES COMPOSITES SUPPLEMENTAIRES (2).....	27
<i>la classe ScrolledComposite</i> .....	27
<i>quelques méthodes de ScrolledComposite</i> .....	27
QUELQUES COMPOSITES SUPPLEMENTAIRES (2).....	28



## Les Dialogs SWT (1)

### les objectifs

- offrir des boîtes de dialogue courantes et génériques
  - messages (information, avertissement, erreur) : **MessageBox**
  - sélection de couleurs : **ColorDialog**, de police de caractère : **FontDialog**
  - sélection de répertoire : **DirectoryDialog**, ouverture-fermeture de fichier : **FileDialog**
  - impression : **PrintDialog**

### les principes généraux

- le parent d'un dialogue est toujours un shell
- les modes d'un dialogue (constantes de la classe SWT) :
  - modal : **APPLICATION\_MODAL**, **PRIMARY\_MODAL**, **SYSTEM\_MODAL**
  - non-modal : **NONE** (par défaut)
- les classes de Dialog SWT NE DOIVENT PAS être sous-classées

### utilisation générique :

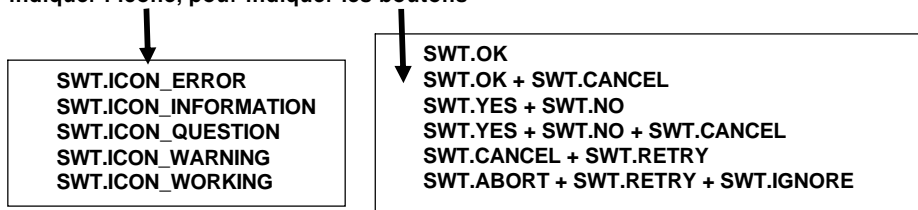
```
<DialogType> dlg = new <DialogType>(shell) ;  
dlg.setSomeData(.....);  
<ReturnType> res = dlg.open();  
if (res == null) { /*l'utilisateur a clické le bouton CANCEL */  
} else { /*traiter la réponse de l'utilisateur */ }
```

Attention !! les **MessageBox** retournent un **int** :  
le test est **if (res == 0) { ..... }**

## Les Dialogs SWT (2)

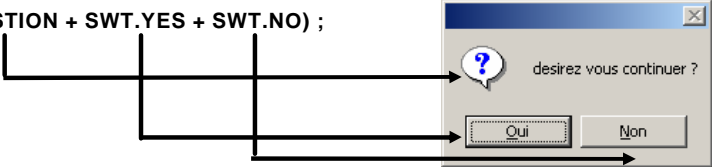
### la boîte de message (la classe MessageBox)

- est formée d'un titre d'un titre + un message + une icône + des boutons
- les styles correspondant sont : **pour indiquer l'icône, pour indiquer les boutons**
- quelques méthodes :  
**void setText(String txt)**  
**void setMessage(String msg)**  
**int open()**

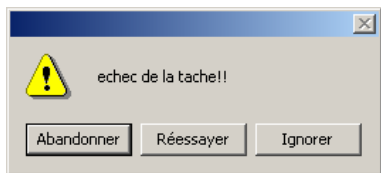


### deux exemples simples

```
dlg1 = new MessageBox(shell, SWT.ICON_QUESTION + SWT.YES + SWT.NO) ;
dlg1.setMessage("desirez vous continuer ?");
int res1 = dlg1.open();
if (res1 == SWT.NO) return ;
.....
```



```
MessageBox dlg2 = new MessageBox(shell, SWT.ICON_WARNING + SWT.ABORT + SWT.RETRY + SWT.IGNORE) ;
dlg2.setMessage("echec de la tache!! ");
int res2 = dlg2.open();
if (res2 == SWT.ABORT) return ;
.....
```



## Les Dialogs SWT (3)

### la sélection des couleurs (la classe ColorDialog)

- pas de style
- retourne une instance de RGB (et non de Color)
- quelques méthodes  
RGB getRGB(), void setRGB( RGB rgb)  
RGB open()

### un exemple simple

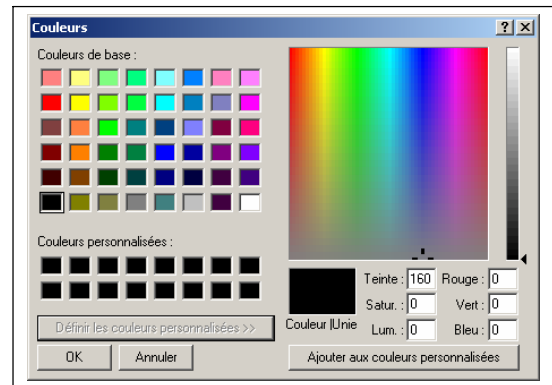
- sélection d'une couleur via un bouton

```
Button bouton = new Button(compo, SWT.PUSH);
```

```
.....
```

```
bouton.setText("couleur");
```

```
bouton.addSelectionListener(new SelectionListener() {  
    public void widgetSelected(SelectionEvent arg0) {  
        ColorDialog cDiag = new ColorDialog(canvas.getShell());  
        RGB rgb = (RGB) cDiag.open();  
        if (rgb != null) {  
            color = new Color(canvas.getDisplay(), rgb);  
            canvas.redraw();  
        }  
    }  
}
```



## Les Dialogs SWT (4)

### la sélection des fichiers pour l'ouverture-fermeture (la classe FileDialog)

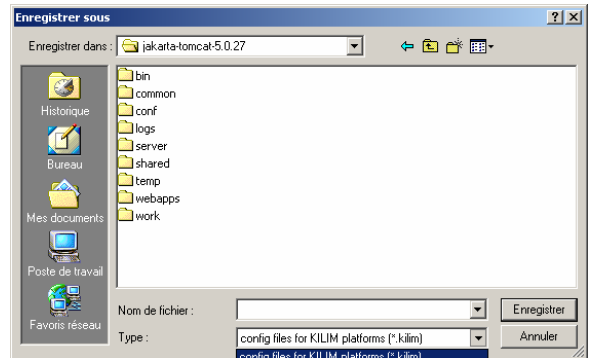
- pas de style
- quelques méthodes
  - String** `getFileName()`, **String[]** `getFileNames()`
  - String[]** `getFilterExtensions()`, **void** `setFilterExtensions(String[] fExt)`
  - String[]** `getFilterNames()`, **void** `setFilterNames(String[] fNm)`
  - String** `getFilterPath()`, **void** `setFilterPath(String fPath)`
  - String** `open()`

### un exemple simple

```

.....
FileDialog filDg = new FileDialog(shell, SWT.MULTI + SWT.SAVE);
filDg.setFilterExtensions(new String[] { "*.kilim", "*.fractal" });
filDg.setFilterNames(new String[] {
    "config files for KILIM platforms (*.kilim)",
    "config files for FRACTAL platforms (*.fractal)" });
filDg.open();
.....

```



### la sélection des répertoires (la classe DirectoryDialog)

- les styles disponibles sont : **SWT.OPEN** (ouverture un seul fichier), **SWT.MULTI** (ouverture un plusieurs fichiers), **SWT.SAVE**
- quelques méthodes  
**String getMessage(), void setMessage(String msg)**  
**String getText(), void setText(String txt)**  
**String getFilterPath(), void setFilterPath(String fPath)**  
**String open()**

### la création de ses propres dialogues

- sous-classer la classe **org.eclipse.swt.widgets.Dialog** (et non les classes concretes)
  - implémenter la méthode **Object open()** [créer la fenêtre, les contrôles, les listeners , ....., afficher la fenêtre]
  - implémenter les getters et setters

### les caractéristiques de la classe List

- ensemble indexé (evt dynamique) d'éléments sélectionnables : items (sous forme de String)
- quelques styles possibles (constantes de la classe SWT)  
**BORDER, H\_SCROLL, V\_SCROLL, READ\_ONLY, MULTI, SINGLE, WRAP**

### quelques méthodes

**void add(String st), void add(String st,int ind)**

**void remove(int index), void removeAll(), remove(int start,int end), void remove(int[] indices), void remove(String itm)**

**void setItem(int index,String itm), void setItems(String[] itms)**

**String getItem(int index), String[] getItems(),int getItemCount()**

**void select(int index), void selectAll(), void select(int start,ind end)**

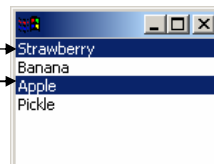
**void deselect(int index), void deselectAll(), void deselect(int start,ind end)**

**int getSelectionIndex(), int[] getSelectionIndices()**

**void showSelection()**

### un exemple simple

```
List list1 = new List(shell, SWT.MULTI | SWT.H_SCROLL);  
list1.setItems(new String[] {"Strawberry","Banana","Apple"});  
list1.add("Pickle");  
list1.setBounds(0,0,60,100);  
.....
```



## Les Combos

### les caractéristiques de la classe Combo

- combinaison de List et de Text
- quelques styles possibles (constantes de la classe SWT) : **DROP\_DOWN**, **SIMPLE**, **READ\_ONLY**
- navigation dans la liste par touche (flèches + 1<sup>ère</sup> lettre)

### quelques méthodes

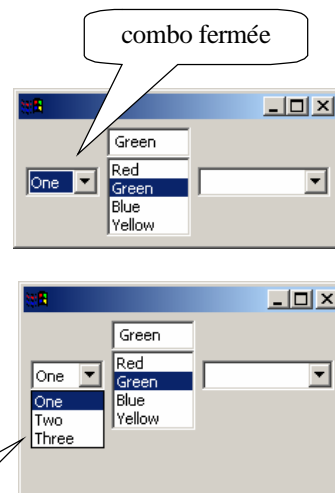
- possèdent les méthodes de List et Text (s'appliquent au champ texte courant)

### un exemple simple

```

Combo combo1 = new Combo(shell, SWT.DROP_DOWN|SWT.READ_ONLY);
combo1.setItems(new String[] {"One", "Two", "Three"});
combo1.select(0);
combo1.setLocation(0,0);
combo1.setSize(100,20);
Combo combo2 = new Combo(shell, SWT.SIMPLE);
combo2.setItems(new String[] {"Red", "Green", "Blue", "Yellow"});
combo2.setBounds(50,50,200,150);
combo2.select(1);
Combo combo3 = new Combo(shell, SWT.DROP_DOWN);
.....

```



combo ouverte

## les caractéristiques de la classe Menu

- menu : ensemble de menu items et (sous) menus
- 3 types de menu :
  - menu correspondant à une barre de menu (placée au sommet de la fenêtre),
  - menu « drop-down » (déroulant),
  - menu « pop-up »
  - les styles correspondant (constantes de la classe SWT) : **BAR**, **DROP\_DOWN**, **POP\_UP**, **NO\_RADIO\_GROUP**
- association d'un menu à un composite, un control ou un menu item via **setMenu()**

## quelques méthodes

**Menu(Decorations parent, int style)** [possibilité de choix : correspond au cas du Shell]  
**Menu(Control parent)** [nécessairement « popup »]  
**Menu(Menu parent)** [nécessairement déroulant]  
**Menu(Menuitem parent)** [nécessairement déroulant]

**MenuItem getItem(int index), MenuItem[] getItems(), MenuItem getParentItem()**  
**int getItemCount()**  
**boolean getEnabled(), void setEnabled(boolean isE)**  
**boolean getVisible(), void setVisible(boolean isV)**  
**Image getImage(), void setImage(image img),**  
**String getText(), void setText(String txt)**

## Les Menus (2)

### un exemple de Menu (associé à la barre de menu)

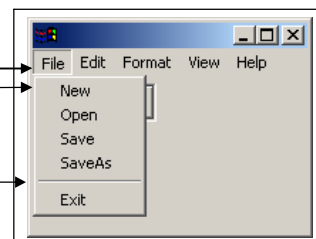
```
Menu menu = new Menu(shell, SWT.BAR);  
shell.setMenuBar(menu);
```

```
MenuItem fileItem = new MenuItem(menu, SWT.CASCADE);  
fileItem.setText("File");
```

```
Menu fileMenu = new Menu(menu);  
fileItem.setMenu(fileMenu);  
MenuItem newItem = new MenuItem(fileMenu, SWT.CHECK);  
newItem.setText("New");
```

```
.....  
new MenuItem(fileMenu, SWT.SEPARATOR);
```

```
.....  
MenuItem exitItem = new MenuItem(fileMenu, SWT.NONE);  
exitItem.setText("Exit");
```

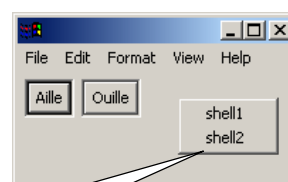
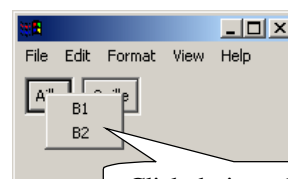


## un exemple de menus pop-up

```
.....  
Button b1 = new Button(shell, SWT.BORDER + SWT.PUSH);  
b1.setText("Aille");  
Menu buttonMenu = new Menu(b1);  
b1.setMenu(buttonMenu);  
MenuItem b1Item = new MenuItem(buttonMenu, SWT.CHECK);  
b1Item.setText("B1");  
MenuItem b2Item = new MenuItem(buttonMenu, SWT.NONE);  
b2Item.setText("B2");
```

```
Button b2 = new Button(shell, SWT.BORDER + SWT.PUSH);  
b2.setText("Ouille");
```

```
Menu shellMenu = new Menu(shell, SWT.POP_UP);  
shell.setMenu(shellMenu);  
MenuItem shellItem1 = new MenuItem(shellMenu, SWT.CHECK);  
shellItem1.setText("shell1");  
MenuItem shellItem2 = new MenuItem(shellMenu, SWT.NONE);  
shellItem2.setText("shell2");  
.....
```



## Les dessins (1)

### la classe GC (Graphical Context)

- noyau du moteur graphique de SWT
- la classe de dessin
- cycle de vie usuel :

```
GC gc = new GC(display) ;
gc.drawXXXX(.....);
gc.drawYYYY(.....);
gc.dispose();
```

### quelques méthodes de la classe GC

```
void setForeground(Color col)
void drawRectangle((int x, int y, int w, int h), void fillRectangle((int x, int y, int w, int h)
void drawOval((int x, int y, int w, int h), void fillOval((int x, int y, int w, int h)
void drawLine(int xOrig, int yOrig, int xDest, int yDest), void drawPolyline(int[ ] points)
void drawPoint(int x, int y)
.....
void drawArc(int xOrig, int yOrig, int width, int height, int begin, int end), void fillArc(.....)
void drawFocus(int x, int y, int width, int height)
void drawText(String txt, int x, int y)
```

### le composite Canvas

- composite spécialement conçu pour contenir des dessins

```
canvas1.addPaintlistener(new Paintlistener() {
    public void paintControl(PaintEvent evt) {
        GC gc = evt.gc ;
        //dessin via gc.drawXXX, gc.fillYYY, ...
        gc.dispose();
    }
});
```

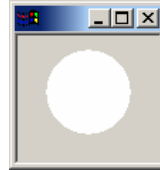
## un exemple simple

```
public class CanvasExample {
    private Canvas canvas;
    private Color color;

    public CanvasExample(Shell aShell) {
        canvas = new Canvas(aShell, SWT.BORDER);
        color = aShell.getDisplay().getSystemColor(SWT.COLOR_WHITE);

        canvas.addPaintListener(new PaintListener() {
            public void paintControl(PaintEvent evt) {
                GC gc = evt.gc;
                gc.setBackground(color);
                gc.fillOval(20, 10, 60, 60);
            }
        });
    }

    public static void main(String[] args) {
        Display display = new Display();
        Shell shell = new Shell(display);
        shell.setLayout(new FillLayout());
        shell.setSize(100,120);
        new CanvasExample(shell);
        shell.open() ;
        .....
    }
}
```



### le Slider

- sélection d'une valeur par un curseur (déplacement du curseur, déplacement par les flèches du clavier, par les flèches du slider, par click de la zone libre)
- les styles possibles (constantes de la classe SWT) : **HORIZONTAL, VERTICAL**
- les principales méthodes sont :  
**int getMinimum(), int getMaximum(), int getIncrement(), getPageIncrement(),  
void setMinimum(int val), void setMaximum(int val), void setIncrement(), void setPageIncrement(int val)  
int getSelection(), void setSelection(int val)**

### la Scrollbar

- identique au Slider mais n'est jamais créé explicitement
- s'obtient par **getHorizontalBar()** et **getVerticalBar()** sur les widgets scrollables (style **SWT.H\_SCROLL, SWT.V\_SCROLL**)

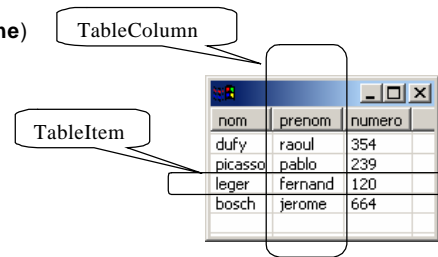
### la Progressbar

- visualisation de l'avancement d'une tâche
- les styles possibles sont les constantes de SWT : **SMOOTH, HORIZONTAL, VERTICAL, INDETERMINATE**
- les méthodes sont :  
**int getMinimum(), int getMaximum()  
void setMinimum(int val), void setMaximum(int val)  
int getSelection(), void setSelection(int val)**

## Les Tables (1)

### les tables (la classe Table)

- une table **Table** est décrit comme un ensemble de **TableColumn** (i.e. colonne) et de **TableItem** (i.e. ligne)
- les types possibles (constantes de la classe SWT) : **SINGLE**, **MULTI**, **CHECK**, **FULL\_SELECTION**, **HIDE\_SELECTION**



### quelques méthodes de la classe Table

**void setLinesVisible(boolean show), void setHeaderVisible(boolean show)**  
**void select(int index), void select(int[] indices), void select(int start, int end), void selectAll()**  
**void deselect(int index), void deselect(int[] indices), void deselect(int start, int end), void deselectAll()**  
**void remove(int index), void remove (int[] indices), void remove (int start, int end), void remove ()**

**TableColumn getColumn(int index), int getColumnCount()**  
**TableItem getItem(int index), TableItem[] getItems()**  
**TableItem[] getSelection(), int getSelectionCount()**  
**void setTopIndex(int index), int getTopIndex()**

**void addSelectionListener(SelectionListener lst), removeSelectionListener(SelectionListener lst)**

sélection d'un item d'une table

## Les Tables (2)

### la classe TableColumn

- correspond à une colonne d'une table
- possède une en-tête qui peut contenir une chaîne de caractère et/ou une image

### quelques méthodes de TableColumn

**void setText(String txt), String getText(), void setImage(Image img), Image getImage()  
void setResizable(boolean isR), boolean getResizable(), void setAlignment(int align), int getAlignement()**

sélection du header d'une colonne

SWT.LEFT, SWT.RIGHT, SWT.CENTER

**void addSelectionListener(SelectionListener lst), void removeSelectionListener(SelectionListener lst)  
void addControlListener(ControlListener lst), void removeControlListener(ControlListener lst)**

redimensionnement d'une colonne

### la classe TableItem

- correspond à une ligne d'une table
- chaque élément peut contenir une chaîne de caractère et/ou une image

### quelques méthodes de TableItem

**void setText(int index, String txt), String getText(index)  
void setImage(int index, Image img), Image getImage(int index)  
void setForeground(int index, Color color), Color getForeground(int index), void setBackground(int index, Color color), ...**

## Les Tables (3)

### un exemple simple

```
String [][] noms = { { "raoul", "dufy"}, {"pablo", "picasso"}, {"fernand", "leger"}, {"jerome", "bosch" } };
String [] numeros = { "354", "239", "120", "664" };
String [] headers = { "nom", "prenom", "numero" };
.....
```

```
aParent.setLayout(new FillLayout());
Table tab1 = new Table(aParent, SWT.MULTI + SWT.FULL_SELECTION);
tab1.setHeaderVisible(true);
tab1.setLinesVisible(true);
```

```
TableColumn cols1[] = new TableColumn[3];
for (int i =0; i < 3; i++) {
    cols1[i] = new TableColumn(tab1, SWT.LEFT);
    cols1[i].setText(headers[i]);
}
}
```

```
tab1.setRedraw(false);
```

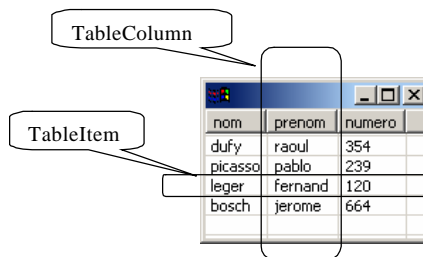
retirer l'autorisation de (re)peindre le tableau

```
for (int i = 0; i < noms.length; i++) {
    TableItem item = new TableItem(tab1, SWT.NONE);
    item.setText(0, noms[i][0]);
    item.setText(1, noms[i][1]);
    item.setText(2, numeros[i]);
}
}
```

Attention. Il faut donner une taille aux colonnes pour faire apparaître le contenu la table !!!

```
for (int i =0; i < 3; i++) cols1[i].pack();
tab1.setRedraw(true);
```

donner l'autorisation de (re)peindre le tableau



## Les Trees (1)

### les arbres (la classe Tree)

- représentation de données hiérarchiques
- arbre SWT = forêt (plusieurs racines possibles)
- arbre **Tree** est décrit comme un ensemble de **Treeltem**
  
- les types possibles sont des constantes de SWT : **SINGLE, MULTI, CHECK**

### quelques méthodes de la classe Tree

```
Treeltem getParentItem(), int getItemCount()  
Treeltem getItem(Point pos), Treeltem[] getItems()  
Treeltem[] getSelection(), int getSelectionCount(), void setSelection(Treeltem[] items)  
void setTopItem(Treeltem item), Treeltem getTopItem()
```

sélection-désélection d'un nœud de l'arbre

```
void addSelectionListener(SelectionListener lst), void removeSelectionListener(SelectionListener lst)  
void addTreeListener(TreeListener lst), void removeTreeListener(TreeListener lst)
```

expansion-collapse d'un nœud de l'arbre

### les nœuds d'un arbre (la classe `TreeItem`)

- modèle classique (nœud parent, nœuds enfants)
- un nœud peut être attaché à un arbre **Tree** (il s'agit alors d'une racine de l'arbre) ou un autre nœud **TreeItem**
- les types possibles sont des constantes de SWT : **SINGLE, MULTI, CHECK**

### quelques méthodes de la classe `TreeItem`

`TreeItem getParentItem()`

`Tree getParent(), TreeItem[] getItems()`

`boolean getExpanded(), void setExpanded(boolean exp)`

`Color getBackground(), void setBackground(Color color), Color getForeground(), void setForeground(Color color)`

`String getText(), void setText(String txt), Image getImage(), void setImage(Image img)`

## Les Trees (3)

## un exemple simple

```

public class SimpleFileTree {
    public static void main(String[] args) {
        .....
        shell.setLayout(new GridLayout(1, false));
        Tree trV = new Tree(shell, SWT.NONE);
        fillTree(trV);
        trV.setLayoutData(new GridData(200, 500));
        .....
    }
    .....
}

```

```

private static void fillTree(Tree aTree) {
    if (aTree == null) return;
    File[] roots = File.listRoots();
    for (int i = 0; i < roots.length; i++) {
        Treeltem trl= new Treeltem(aTree, SWT.NONE);
        trl.setText(roots[i].getAbsolutePath());
        fillTreeltem(trl, roots[i], 0);
    }
    aTree.setVisible(true);
}

```

```

private static void fillTreeltem(Treeltem altem, File aFile, int aLevel) {
    if (aLevel > 5) return;
    if (altem == null || aFile == null) return;
    File[] files = aFile.listFiles();
    if (files == null) return;
    for (int i = 0; i < files.length; i++) {
        Treeltem trl = new Treeltem(altem, SWT.NONE);
        trl.setText(files[i].getName());
        fillTreeltem(trl, files[i], aLevel + 1);
    }
}

```

## Les dossiers tabulés (1)

### quelques généralités

- un dossier **TabFolder** est ensemble de **TabItem**
- les styles possibles pour **TabFolder** sont donnés par des constantes de **SWT** : **TOP** (valeur par défaut), **BOTTOM**

- création d'un dossier tabulé

**Attention : parent doit posséder un Layout. Sinon tabs invisibles**

```
.....  
TabFolder tFolder = new TabFolder(parent, SWT.NONE) ;  
TabItem tItem1 = new TabItem(tFolder, SWT.NONE);  
.....
```

### quelques méthodes de TabFolder :

```
void addSelectionListener(SelectionListener ln), void removeSelectionListener(SelectionListener ln)  
TabItem getItem(int index), TabItem[] getItems(), int getItemCount()  
void setSelection(int index), void setSelection(TabItem[] items), TabItem[] getSelection()
```

### quelques méthodes de TabItem :

```
Control getControl(), void setControl(Control ctl)  
Image getImage(), void setImage(), String getText(), void setText(String txt)  
ToolTipText getToolTipText(), void setToolTipText(ToolTipText ttt)  
TabItem getItem(int index), TabItem[] getItems(), int getItemCount()  
void setSelection(int index), void setSelection(TabItem[] items), TabItem[] getSelection()
```

## Les dossiers tabulés (2)

### un exemple simple

```

.....
shell.setLayout(new FillLayout());
TabFolder tFolder = new TabFolder(shell, SWT.TOP);

TabItem tItem = new TabItem(tFolder,SWT.NONE);
tItem.setText("module");
tItem.setToolTipText("ceci est une tab pour la definition des modules");
Composite compo = new Composite(tFolder, SWT.NONE);
compo.setLayout(new GridLayout(3, false));
tItem.setControl(compo);
for (int i = 0; i < 5; i++) {
    Label labl = new Label(compo, SWT.HORIZONTAL + SWT.BORDER);
    labl.setText("module numero " + i);
}

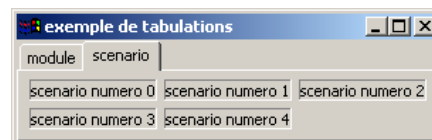
tItem = new TabItem(tFolder,SWT.NONE);
tItem.setText("scenario");
tItem.setToolTipText("ceci est une tab pour la definition des scenarii");
Composite compo = new Composite(tFolder, SWT.NONE);
compo.setLayout(new GridLayout(2, false));
for (int i = 0; i < 5; i++) {
    Label labl = new Label(compo, SWT.HORIZONTAL + SWT.BORDER);
    labl.setText("scenario numero " + i);
}
.....

```

création du dossier

création d'un item

association du contrôle à l'item



## Quelques composites supplémentaires (1)

### la classe SashForm

- un composite séparé en plusieurs parties par un sash mobile
- les styles possibles (constantes de la classe **SWT** : **HORIZONTAL** et **VERTICAL**)
- par défaut partage en parties égales (méthode `setWeights(new int[] { poidsRelatif1, poidsRelatif2 })`)

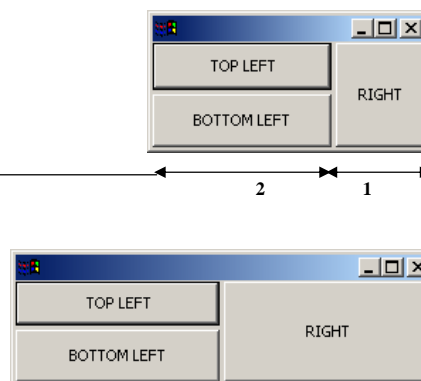
### quelques méthodes de SashForm

`void setBackground(Color color), void setForeground(Color color), Color getBackground(), ...`  
`void setWeights(int[] relW), int[] getWeights()`

### un exemple simple

```

.....
shell.setLayout(new FillLayout());
SashForm sForm1 = new SashForm(shell, SWT.HORIZONTAL);
SashForm sForm1L = new SashForm(sForm1, SWT.VERTICAL);
Button but1 = new Button(sForm1L, SWT.PUSH);
but1.setText(" TOP LEFT");
Button but2 = new Button(sForm1L, SWT.PUSH);
but2.setText("BOTTOM LEFT");
Button but3 = new Button(sForm1, SWT.PUSH);
but3.setText("RIGHT");
sForm1.setWeights(new int[] { 2, 1 });
.....
    
```



## Quelques composites supplémentaires (2)

### la classe ScrolledComposite

- un composite muni de scrollbars
- les styles possibles sont des constantes de SWT : **H\_SCROLL**, **SWT.V\_SCROLL**
- ne possède qu'UN SEUL ENFANT !!!!

### quelques méthodes de ScrolledComposite

**void setContent(Composite fils)**

**void setMinimumSize(int width, int height)**

ABSOLUMENT nécessaire pour un comportement correct du ScrolledComposite

La taille minimale contraint la taille de l'enfant du ScrolledComposite  
Elle contrôle l'apparition des barres de scroll

**void setExpandHorizontal(boolean allow), void setExpandVertical(boolean allow)**

contrôle le comportement de l'enfant du ScrolledComposite

**void setOrigin(int x, int y), void setOrigin(Point orig), point getOrigin()**

**void setAlwaysShowScrollBars(boolean show), boolean getAlwaysShowScrollBars()**

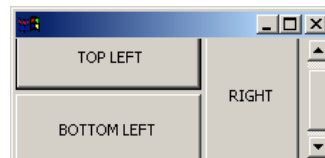
## Quelques composites supplémentaires (2)

### un exemple simple

```

.....
shell.setLayout(new FillLayout());
ScrolledComposite sCompo1 = new ScrolledComposite(aPar, SWT.H_SCROLL + SWT.V_SCROLL);
sCompo1.setContent(compo);
sCompo1.setMinimumSize(400,300);
sCompo1.setExpandHorizontal(false);
sCompo1.setExpandVertical(false);

```



```

Composite compo = new Composite(sCompo1, SWT.NONE);
compo.setLayout(new FillLayout());

```

```

SashForm sForm1 = new SashForm(compo, SWT.HORIZONTAL);
SashForm sForm1L = new SashForm(sForm1, SWT.VERTICAL);
Button but1 = new Button(sForm1L, SWT.PUSH);
but1.setText(" TOP LEFT");
Button but2 = new Button(sForm1L, SWT.PUSH);
but2.setText("BOTTOM LEFT");
.....

```

