Eclipse RCP Part VI

Automotive Financial Services Insurance Life Science & Healthcare Public Sector Telecommunications & Media Travel & Logistics Utilities

.consulting .solutions .partnership

.msg systems

Objective

- Building a SWT GUI
- Learning to integrate SWT snippets

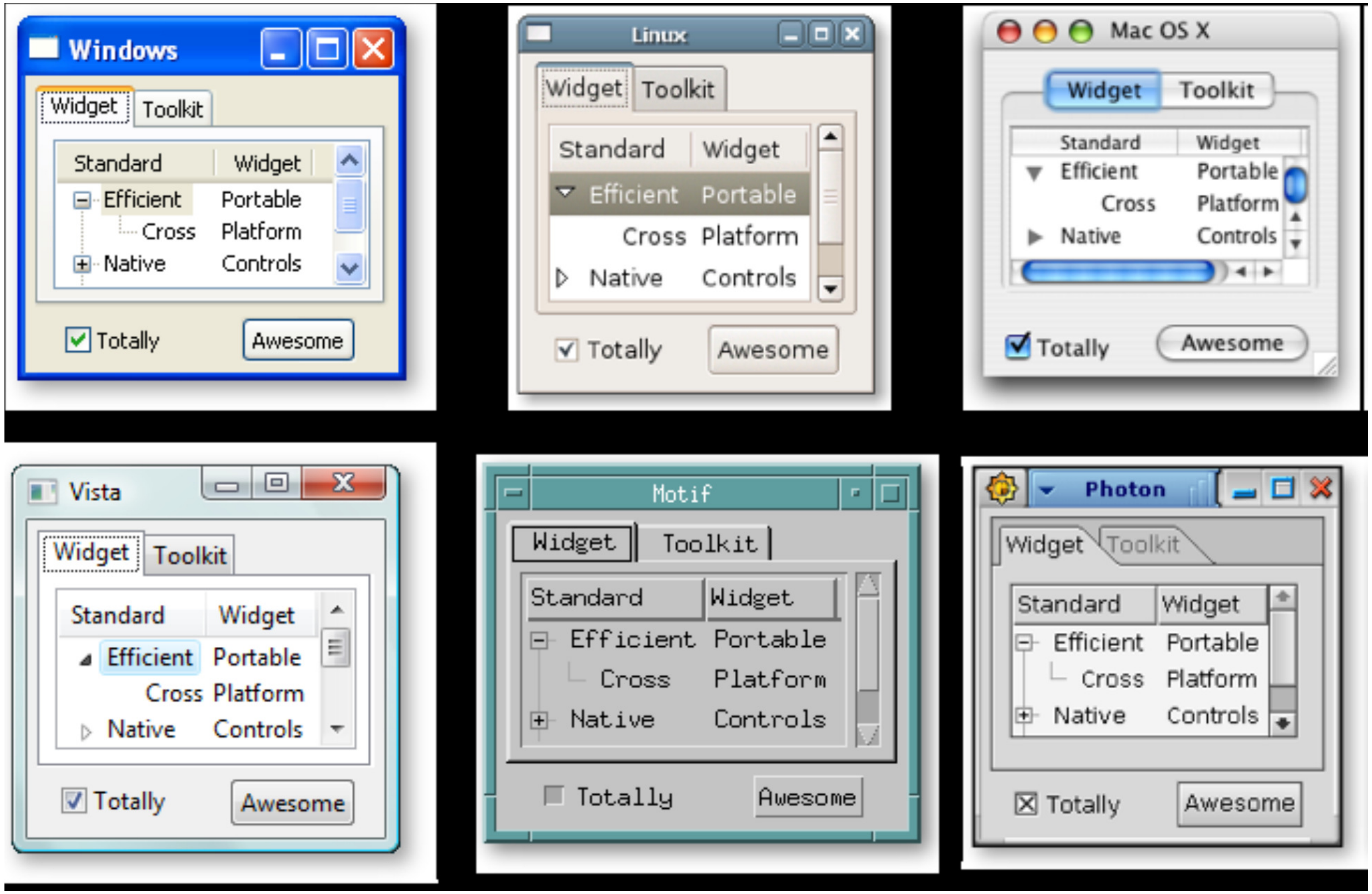
SWT

- OS like look & feel
- SWT provides access to native operating system widgets using a Java API.
- Some OS Functionality is emulated
- SWT provides a low level abstraction
- JFace provides a higher level of abstraction

Problem:

- Platform dependent behavior - it is highly recommend to test on each platform you want to support with your application

Native look & feel



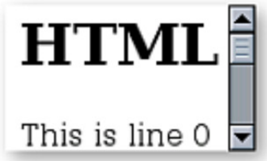
Low level abstraction with SWT

```
public void createPartControl(Composite parent) {
    parent.setLayout(new GridLayout());
    Table table = new Table (parent, SWT.MULTI | SWT.BORDER | SWT.FULL_SELECTION);
    table.setLinesVisible (true);
    table.setHeaderVisible (true);
    GridData data = new GridData(SWT.FILL, SWT.FILL, true, true);
    data.heightHint = 200;
    table.setLayoutData(data);
    String[] titles = {" ", "C", "!", "Description", "Resource", "In Folder", "Loca
    for (int i=0; i<titles.length; i++) {
        TableColumn column = new TableColumn (table, SWT.NONE);
        column.setText (titles [i]);
    }
    int count = 128;
    for (int i=0; i<count; i++) {
        TableItem item = new TableItem (table, SWT.NONE);
        item.setText (0, "x");
        item.setText (1, "y");
        item.setText (2, "!");
        item.setText (3, "this stuff behaves the way I expect");
        item.setText (4, "almost everywhere");
        item.setText (5, "some.folder");
        item.setText (6, "line " + i + " in nowhere");
    }
    for (int i=0; i<titles.length; i++) {
        table.getColumn (i).pack ();
    }
    parent.pack ();
```

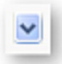
Higher level abstraction with JFace

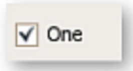
```
public void createPartControl(Composite parent) {
    viewer = new TableViewer(parent, SWT.MULTI | SWT.H_SCROLL | SWT.V_SCROLL);
    viewer.setContentProvider(new ViewContentProvider());
    viewer.setLabelProvider(new ViewLabelProvider());
    viewer.setSorter(new NameSorter());
    viewer.setInput(getViewSite());
```

Widgets

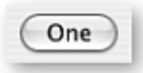


Browser
javadoc - snippets

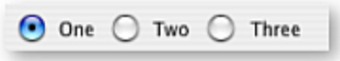

Button (`SWT.ARROW`)
javadoc - snippets



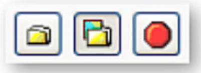
Button (`SWT.CHECK`)
javadoc - snippets

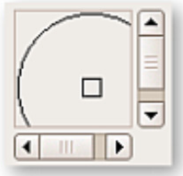


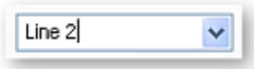
Button (`SWT.PUSH`)
javadoc - snippets



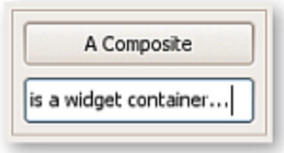
Button (`SWT.RADIO`)
javadoc - snippets


Button (`SWT.TOGGLE`)
javadoc - snippets


Canvas
javadoc - snippets

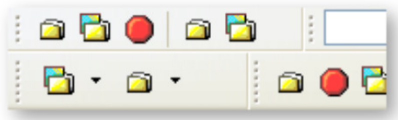


Combo
javadoc - snippets

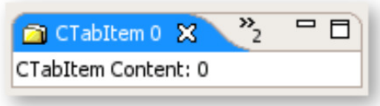


Composite
javadoc - snippets

Widgets II

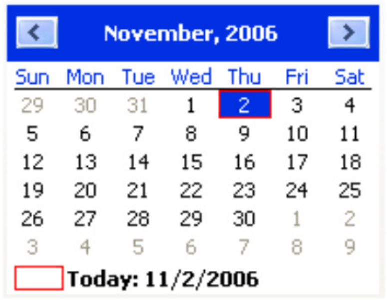


CoolBar
javadoc - snippets

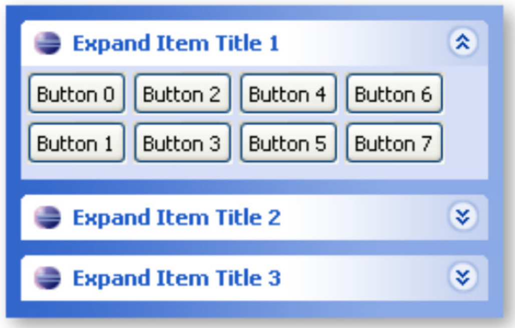


CTabItem Content: 0

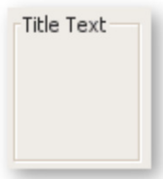
CTabFolder
javadoc - snippets



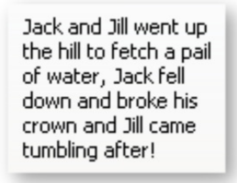
DateTime
javadoc - snippets



ExpandBar
javadoc - snippets



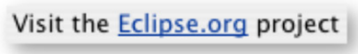
Group
javadoc



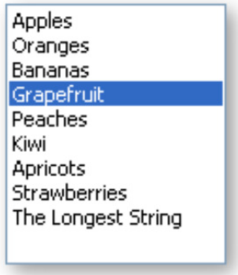
Jack and Jill went up the hill to fetch a pail of water, Jack fell down and broke his crown and Jill came tumbling after!

Label
javadoc - snippets

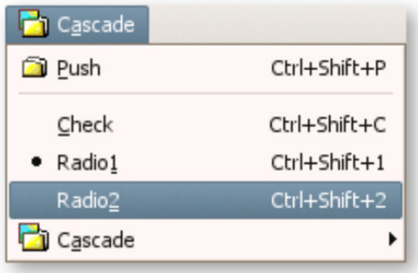
Widgets III



Link
javadoc - snippets

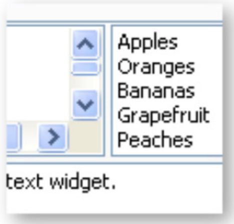


List
javadoc - snippets

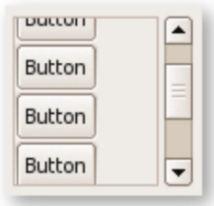


Menu
javadoc - snippets

ProgressBar
javadoc - snippets



Sash
javadoc - snippets



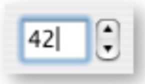
ScrolledComposite
javadoc - snippets

Shell
javadoc - snippets

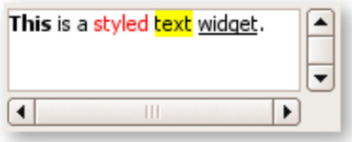
Slider
javadoc - snippets

Scale
javadoc - snippets

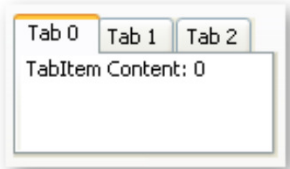
Widget IV



Spinner
javadoc - snippets

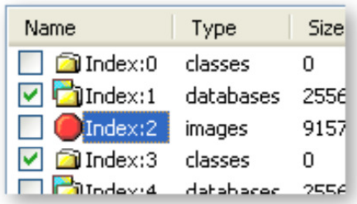

This is a styled text widget.

StyledText
javadoc - snippets



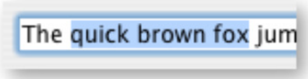
TabItem Content: 0

TabFolder
javadoc - snippets

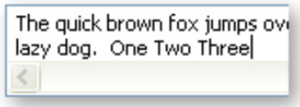


Name	Type	Size
☐ Index:0	classes	0
✓ Index:1	databases	2556
☐ Index:2	images	9157
✓ Index:3	classes	0
☐ Index:4	databases	2556

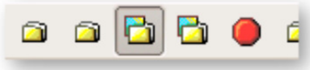
Table
javadoc - snippets


The quick brown fox jum

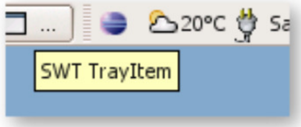
Text (`SWT.SINGLE`)
javadoc - snippets


The quick brown fox jumps ov
lazy dog. One Two Three

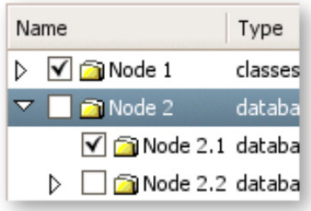
Text (`SWT.MULTI`)
javadoc - snippets



ToolBar
javadoc - snippets



Tray
javadoc - snippets

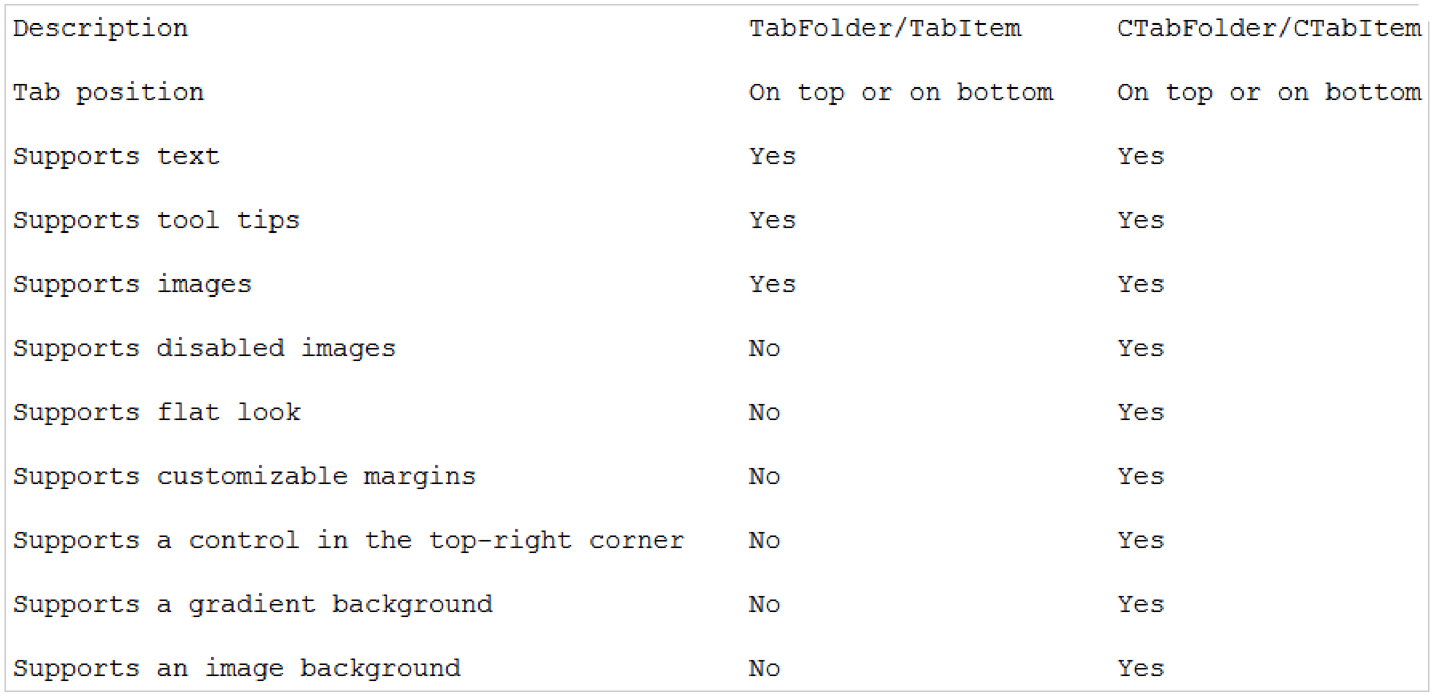


Name	Type
▷ ✓ Node 1	classes
▽ ☐ Node 2	databa
✓ Node 2.1	databa
▷ ☐ Node 2.2	databa

Tree
javadoc - snippets

CTabFolder versus TabFolder

CTabFolder and CTabItem add flexibility to the standard tabs.



Description	TabFolder/TabItem	CTabFolder/CTabItem
Tab position	On top or on bottom	On top or on bottom
Supports text	Yes	Yes
Supports tool tips	Yes	Yes
Supports images	Yes	Yes
Supports disabled images	No	Yes
Supports flat look	No	Yes
Supports customizable margins	No	Yes
Supports a control in the top-right corner	No	Yes
Supports a gradient background	No	Yes
Supports an image background	No	Yes

SWT Control

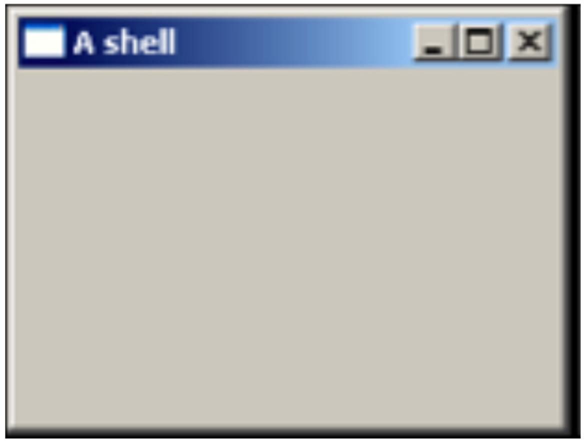
- The control is a subclass of Widget and acts as a base class for all "windowed" UI classes.
- A control...
  - knows it's size and position
  - can be enabled or disabled
  - can be shown or hidden (marked as visible or invisible)
  - can receive keyboard input focus
  - can handle focus, mouse, keyboard, size and paint events

SWT Composite

- Composite is a special control that contains other controls
  - Composite manages it's child elements
  - When a composite is disposed using the dispose() method, all related children will be disposed recursively
  - Supports layouts
  - Composite subclasses: Group, Canvas, Shell

SWT Shell

- Shell is a special subclass of Composite and represents the "window" concept of the underlying graphical system
- There can be primary (top level) and secondary windows (main window vs. dialog window)
- For a GUI at least one top level Shell (window) is required
- In Eclipse RCP the top level Shell is created by the framework



The parent widget

- All widgets are arranged in a tree-like structure
- Composite is a container on which controls can be added
- Every widget (except for top-level shell) has one widget as parent

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

Display display = new Display();
Shell parent = new Shell (display);

Label label = new Label (parent, SWT.NONE);
label.setText ("Enter your name:");

Text text = new Text (parent, SWT.BORDER);
Button ok = new Button (parent, SWT.PUSH);
```

Parent

Style Flags

createPartControl

SWT Standalone App

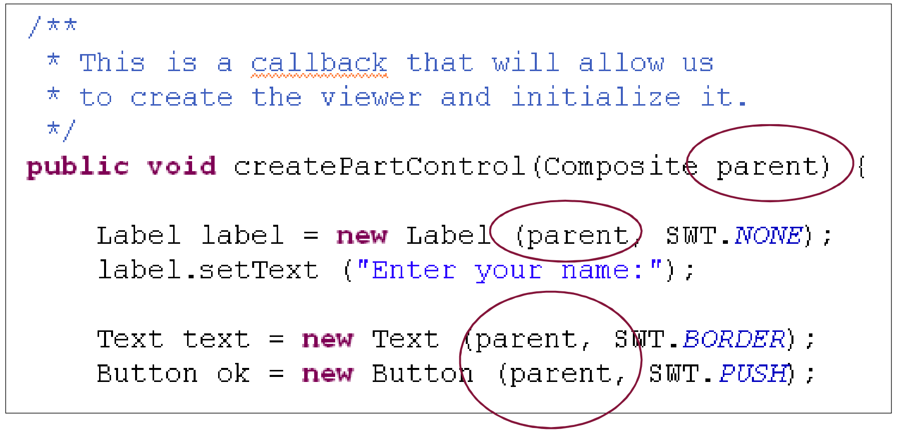
~~public static void main (String [] args) {~~

~~Display display = new Display();~~
~~Shell **shell** = new Shell (display);~~

Label label = new Label (**shell**, SWT.NONE);
label.setText ("Enter your name:");

Text text = new Text (**shell**, SWT.BORDER);
Button ok = new Button (**shell**, SWT.PUSH);

Eclipse View



```
/**
 * This is a callback that will allow us
 * to create the viewer and initialize it.
 */
public void createPartControl(Composite parent) {

    Label label = new Label (parent, SWT.NONE);
    label.setText ("Enter your name:");

    Text text = new Text (parent, SWT.BORDER);
    Button ok = new Button (parent, SWT.PUSH);
```

Lab

- Open http://www.eclipse.org/swt/snippets/
- copy some of the snippets to your views (createPartControl(…))
- adapt the code to work with the new parent control

Layout basics

- By default SWT does not set size or position of it's components.
- Every new control has the size (0,0), so it is invisible.
- Applications can define positions and sizes of controls when they are created or later (inside resize listener):
  - Control.setSize(Point point) or others like Control.setSize(int x, int y)

- Alternatively, a Layout may be specified. An instance of the Layout class will be responsible for sizing and positioning controls.

Layout and LayoutData

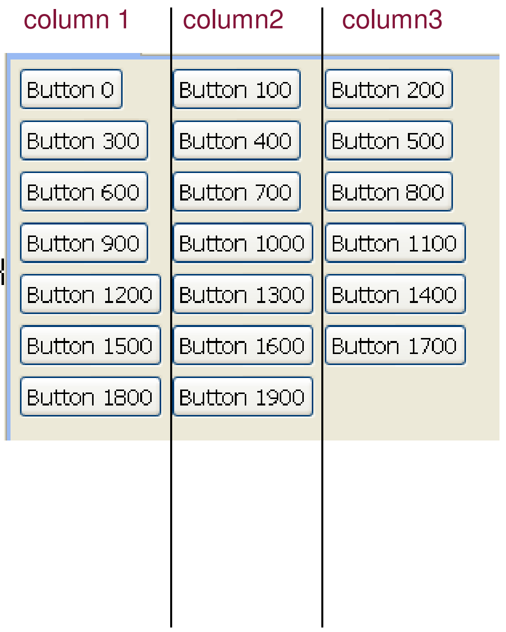
- A layout controls the position and size of child elements in a Composite
- Composite.setLayout(Layout layout)

- The size and positioning of a control can be defined by setting an object with layout data
- Control.setLayoutData(Object layoutData)

- SWT layouts are similar to layouts in AWT / Swing

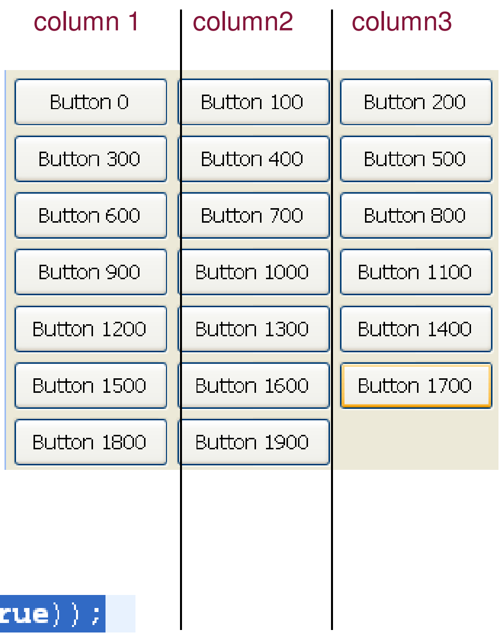
GridLayout

```
/**
 * This is a callback that will allow us
 * to create the viewer and initialize it.
 */
public void createPartControl(Composite parent) {

    GridLayout layout = new GridLayout();
    layout.numColumns = 3;
    parent.setLayout(layout);

    for (int i = 0; i < 20; i++) {
        Button b = new Button(parent, SWT.PUSH);
        b.setText("Button "+i*100);
    }
}
```

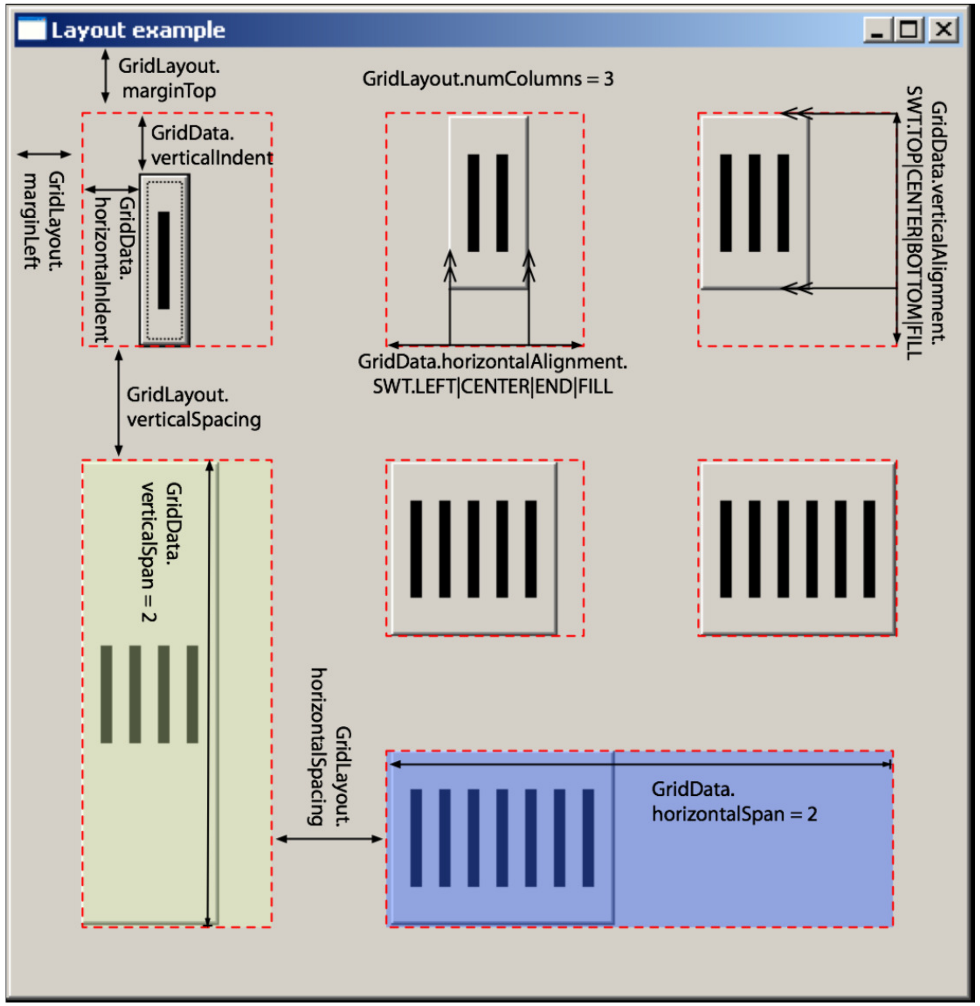


setLayoutData

```
/**
 * This is a callback that will allow us
 * to create the viewer and initialize it.
 */
public void createPartControl(Composite parent) {

    GridLayout layout = new GridLayout();
    layout.numColumns = 3;
    parent.setLayout(layout);

    for (int i = 0; i < 20; i++) {
        Button b = new Button(parent, SWT.PUSH);
        b.setLayoutData(new GridData(SWT.FILL, SWT.FILL, true, true));
        b.setText("Button "+i*100);
    }
}
```



GridLayout & GridData

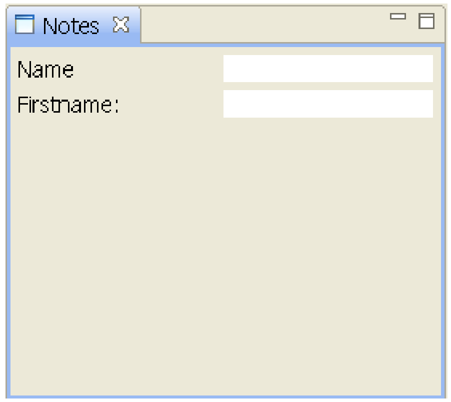


Example

```
/**
 * This is a callback that will allow us
 * to create the viewer and initialize it.
 */
public void createPartControl(Composite parent) {

    GridLayout layout = new GridLayout();
    layout.numColumns = 2;
    parent.setLayout(layout);

    Label label = new Label(parent,SWT.NULL);
    label.setText("Name");
    label.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));

    Text text = new Text(parent,SWT.NULL);
    text.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
```

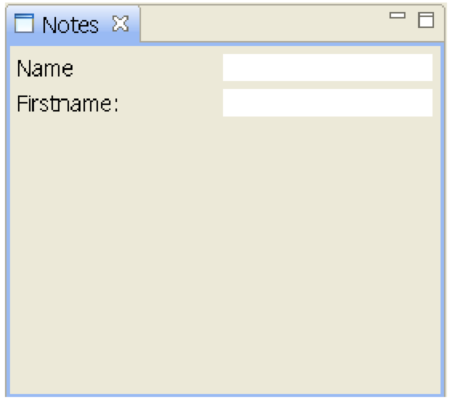
Example

```
/**
 * This is a callback that will allow us
 * to create the viewer and initialize it.
 */
public void createPartControl(Composite parent) {

    GridLayout layout = new GridLayout();
    layout.numColumns = 2;
    parent.setLayout(layout);

    Label label = new Label(parent,SWT.NULL);
    label.setText("Name");
    label.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));

    Text text = new Text(parent,SWT.NULL);
    text.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));

    Label label2 = new Label(parent,SWT.NULL);
    label2.setText("Firstname:");
    label2.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));

    Text text2 = new Text(parent,SWT.NULL);
    GridData gd2 = new GridData(GridData.FILL_HORIZONTAL);
    text2.setLayoutData(gd2);
```



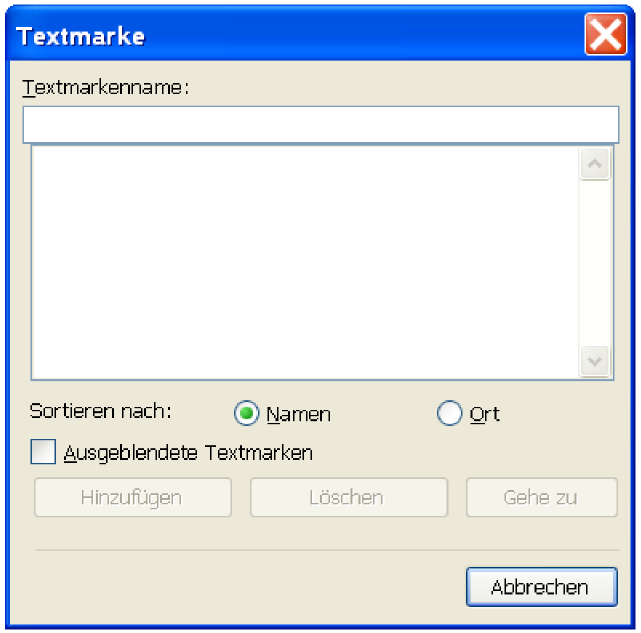
Lab

- Create some views of your own


Notes

Name

Firstname:


Textmarke

Textmarkenname:

Sortieren nach: [✓ Namen] Ort

Ausgeblendete Textmarken

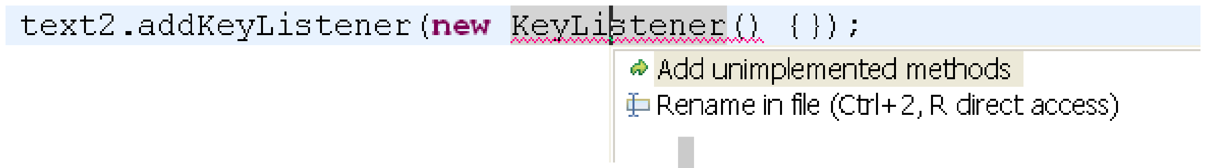
Listener

- Typical SWT listeners which can be attached to a control are:
  - SelectionListener
  - KeyListener
  - MouseListener and others (see org.eclipse.swt.events package)
- For most listeners appropriate adapter-classes with empty implementations of required methods exist, e.g.:
- MouseListener → MouseAdapter
- SelectionListener → SelectionAdapter

Tips & Tricks

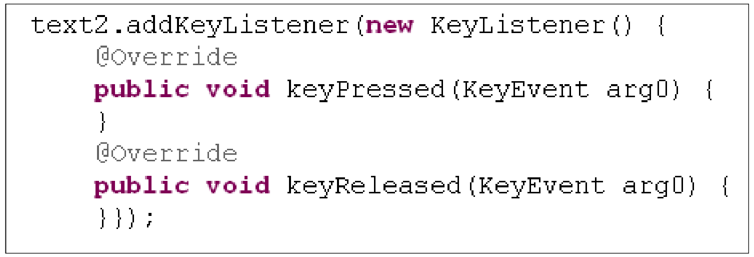
- Enter

  text2.addKeyListener(**new** KeyListener() {});
  and then use the content assist



```
text2.addKeyListener(new KeyListener() {});
```

Add unimplemented methods
Rename in file (Ctrl+2, R direct access)



```
text2.addKeyListener(new KeyListener() {
    @Override
    public void keyPressed(KeyEvent arg0) {
    }
    @Override
    public void keyReleased(KeyEvent arg0) {
    }});
```

Vielen Dank für Ihre Aufmerksamkeit

.consulting .solutions .partnership

.msg systems