

Eclipse 开发经典教程：SWT 布局

在 Java 中，GUI 程序开发的目标之一是跨平台，而每种类型操作系统对屏幕的定义不一样，所以 Swing 中引入了布局的概念，对子组件的位置和大小等信息进行定义。SWT 中也采用了布局方式，用户可使用布局来控制组件中元素的位置和大小等信息。

另外，如果组件没有设置布局信息，也可以通过坐标进行相对定位。在介绍 Control 类的时候，其中有一个方法 “setBounds (int x, int y, int width, int height)” 可以指定组件相对于父组件的位置和组件的大小。在这种方式下，父组件大小和其他信息的改变不会对当前组件有影响。复合组件常常包含多个控件，可以使用以下两种方法安排这些控件。

(1) 绝对定位：为每个控件设置明确的 X 和 Y 位置 (setBounds)，并通过代码设置一定的宽度和高度。

(2) 托管定位：每个控件的 X、Y、宽度和高度都是通过 LayoutManager 设置的。

在多数情况下，应该选择使用 LayoutManagers，因为可以很容易地调整它们来适应可变大小的 GUI。SWT 提供了一些常用的布局管理器供用户使用；在布局管理器中，每当重新设置复合组件的大小时，都需要进行定位。

布局管理器常常是专为某一个复合组件设计的。一些布局管理器只使用它们自身的参数就可以控制，而另一些布局管理器还需要其他参数 (LayoutData)，该参数是在设置布局管理器的复合组件中的每个控件上指定的。SWT 中常用的布局管理器有如下一些。

● **FillLayout**：子组件将以相同的大小填充到父组件中。[λ](#)

● **RowLayout**: 子组件将在父组件上一行或几行显示（设置相应的属性值，子组件会自动换行）。λ

● **GridLayout**: 网格格式布局，子组件可以指定占用父组件中几个格，以及组件填充哪几个网格。λ

● **FormLayout**: 可以通过 **FormAttachment** 以父组件或子组件的边作为相对位置，进行精确布局。λ

为组件添加布局信息的步骤如下。

1. 创建布局（Layout）类。
2. 通过窗口组件的 `setLayout` 方法设置相应的布局类。
3. 设置子组件的布局信息。

当窗口组件设置了布局信息后，窗口组件显示的时候将会调用相应的布局类对窗口组件的子组件进行布局、定位和计算子组件大小的操作，从而使窗口组件以更好的方式显示在父组件中。下面将介绍 Eclipse 中提供的几种常用的布局方式。

FillLayout 布局

FillLayout 是非常简单的一种布局方式，它会以同样大小对父组件中的子组件进行布局，这些子组件将以一行或一列的形式排列。

一般来说，用户可以在任务栏、工具栏中放置 **FillLayout** 布局，通过 **FillLayout** 布局对子组件进行定位，也可以当子组件只有一个组件时，通过 **FillLayout** 布局填充整个父组件的空间。

FillLayout 的风格

FillLayout 布局中，可以把子组件按水平或垂直的方式进行排列，这些风格是当创建 **FillLayout** 实类时以参数形式指定的，如表 1 所示。

表 1 FillLayout 的风格

	初 始 状 态	调整父窗口大小后的状态
SWT.HORIZONT AL (default)		
SWT.VERTICAL		

FillLayout 布局实例

FillLayout 是简单而且很常用的布局，下面通过实例展示 FillLayout 的布局效果，代码如例程 1 所示。

例程 1 FillLayoutSample.java

```
/**
```

```
* 为了节省篇幅，所有的 import 类已经被注释
```

```
* 读者可以通过 ctrl+shift+o 快捷键，自动引入所依赖的类
```

```
* 如果有问题可发邮件到 ganshm@gmail.com
```

```
* */
```

```
public class FillLayoutSample {
```

```
Display display = new Display();
```

```
Shell shell = new Shell(display);
```

```
public FillLayoutSample() {
```

```
//新建 FillLayout 布局，设置子组件与水平方式排列
```

```
FillLayout fillLayout = new FillLayout(SWT.HORIZONTAL);
```

```
//指定子组件的上、下边距为多少像素
```

```
fillLayout.marginHeight = 25;
```

```
//指定子组件的左、右边距为多少像素
```

```
fillLayout.marginWidth = 25;
```

```
//指定子组件之间距离为多少像素
```

```
fillLayout.spacing = 10;
```

```
//设定父组件的布局方式
```

```
shell.setLayout(fillLayout);
```

```
Button button1 = new Button(shell, SWT.PUSH);
```

```
button1.setText("button1");
```

```
Button button2 = new Button(shell, SWT.PUSH);
```

```
button2.setText("button number 2");
```

```
Button button3 = new Button(shell, SWT.PUSH);
```

```
button3.setText("3");
```

```
shell.pack();
```

```
shell.open();
```

```
while (!shell.isDisposed()) {
```

```
if (!display.readAndDispatch()) {
```

```
display.sleep();
```

```
}
```

```
}
```

```
display.dispose();
```

```
}
```

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

```
new FillLayoutSample();
```

```
}
```

```
}
```

程序中通过 `marginHeight`、`marginWidth` 和 `spacing` 指定了边距和子组件的间距，程序运行效果如图 1 所示。



图 1 FillLayout 布局实例

RowLayout 布局

相对于 FillLayout 来说，RowLayout 比较灵活，功能也比较强。用户可以设置布局中子元素的大小、边距、换行及间距等属性。


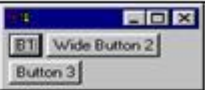

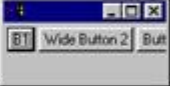

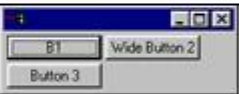





RowLayout 的风格

RowLayout 中可以相关的属性设定布局的风格，用户可以通过“RowLayout.属性”的方式设置 RowLayout 的布局风格，RowLayout 中常用的属性如下。

- Wrap: 表示子组件是否可以换行（true 为可换行）。λ
- Pack: 表示子组件是否为保持原有大小（true 为保持原有大小）。λ
- Justify: 表示子组件是否根据父组件信息做调整。λ
- MarginLeft: 表示当前组件距离父组件左边距的像素点个数。λ
- MarginTop: 表示当前组件距离父组件上边距的像素点个数。λ
- MarginRight: 表示当前组件距离父组件右边距的像素点个数。λ
- MarginBottom: 表示当前组件距离父组件下边距的像素点个数。λ
- Spacing: 表示子组件之间的间距像素点个数。λ

另外，RowLayout 可以通过 RowData 设置每个子组件的大小，例如“button.setLayoutData (new RowData(60, 60))” 将设置 buton 的大小为 (60,60)，RowLayout 风格如表 2 所示。

表 2 RowLayout 风格

	初 始 状 态	调整大小后的状态
<code>wrap = true</code> <code>pack = true</code> <code>justify = false</code> <code>type = SWT.HORIZONTAL</code> (defaults)		
<code>wrap = false</code> (clips if not enough space)		
<code>pack = false</code> (all widgets are the same size)		
<code>justify = true</code> (widgets are spread across the available space)		
<code>type = SWT.VERTICAL</code>  are arranged bitsCNi.comms)		

RowLayout 布局实例

RowLayout 是很常用的布局，而且不太复杂，下面通过实例展示 RowLayout 的布局效果，代码如例程 2 所示。

例程 2 RowLayoutExample.java

```
public class RowLayoutExample {  
  
    Display display;  
  
    Shell shell;  
  
    RowLayoutExample() {
```

```
display = new Display();
```

```
shell = new Shell(display);
```

```
shell.setSize(250, 150);
```

```
shell.setText("A RowLayout Example");
```

```
//新建 RowLayout 布局
```

```
RowLayout rowLayout = new RowLayout();
```

```
//子组件保持原有大小
```

```
rowLayout.pack = true;
```

```
//子组件可换行
```

```
rowLayout.wrap = true;
```

```
//根据父组件信息调整位置
```

```
rowLayout.justify = true;
```

```
//左边距为 30 像素
```

```
rowLayout.marginLeft = 30;
```

```
//上边距为 30 像素
```

```
rowLayout.marginTop = 30;
```

```
//设定父组件 RowLayout 布局
```

```
shell.setLayout(rowLayout);
```

```
final Text t = new Text(shell, SWT.SINGLE | SWT.BORDER);
```

```
final Button b = new Button(shell, SWT.BORDER);
```

```
final Button b1 = new Button(shell, SWT.BORDER);
```

```
//设置子组件大小
```

```
b1.setLayoutData(new RowData(60, 60));
```

```
b.setText("OK");
```

```
b1.setText("Cancel");
```

```
shell.open();
```

```
while (!shell.isDisposed()) {
```

```
if (!display.readAndDispatch())
```

```
display.sleep();
```

```
}
```

```
display.dispose();
```

```
}
```

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

```
new RowLayoutExample();
```

```
}
```

```
}
```

程序中指定了边距和子组件的间距，以及子组件大小的信息，程序运行效果如图 2 所示。



图 2 RowLayout 布局实例

GridLayout 布局

GridLayout 布局的功能非常强大，也是笔者常用的一种布局方式。GridLayout 是网格格式布局，它把父组件分成一个表格，默认情况下每个子组件占据一个单元格的空间，每个子组件按添加到父组件的顺序排列在表格中。

GridLayout 提供了很多的属性，可以灵活设置网格的信息。另外，GridLayout 布局提供了 GridData 类，子组件可以设置相应的 GridData，例如 “dogPhoto.setLayoutData(gridData)”，GridData 可以设置每个组件当做单元格的信息。

GridLayout 的风格

GridLayout 类提供了 GridLayout 布局中划分网格的信息，主要通过以下几个参数进行设置。

NumColumns: 通过 “gridLayout.numColumns” 属性可以设置父组件中分几列显示子组件，如表 3 所示。 λ

表 3 NumColumns 效果

列 数	显 示 效 果
<code>numColumns = 1</code>	
<code>numColumns = 2</code>	
 <code>numColumns = 3</code>	

MakeColumnsEqualWidth: 通过 “gridLayout. makeColumnsEqualWidth” 属性可以设置父组件中子组件是否有相同的列宽，当 MakeColumnsEqualWidth 为 true 时表示每列的列宽相等。

●MarginLeft: 表示当前组件距离父组件左边距的像素点个数。 λ

●MarginRight: 表示当前组件距离父组件右边距的像素点个数。 λ

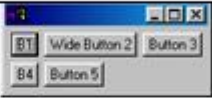
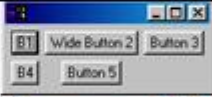


- MarginTop: 表示当前组件距离父组件上边距的像素点个数。 λ
- MarginBottom: 表示当前组件距离父组件下边距的像素点个数。 λ
- HorizontalSpacing: 表示子组件的水平间距。 λ
- VerticalSpacing: 表示子组件的垂直间距。 λ

GridData 的相关属性

GridLayout 布局的灵活之处在于它利用网格布局数据 GridData。通过 GridData 可以设置子组件在网格中的填充方式、大小边距等信息，用户可以通过子组件的 setLayoutData 方法设置网格布局数据。

GridData 可以控制子组件在网格中的位置大小等相关显示信息。GridData 可以设置如下的一些属性。

表 4 组件水平对齐方式

HorizontalAlignment 的值	显示效果
horizontalAlignment = GridData.BEGINNING (default)	
horizontalAlignment = GridData.CENTER	
horizontalAlignment = GridData.END	
horizontalAlignment = GridData.FILL	

- HorizontalAlignment: 表示水平对齐方式。水平对齐方式有如下几种，如表 4 所示，其中“Button5”按钮显示了水平对齐的方式。
- VerticalAlignment: 表示子组件的垂直对齐方式，值和水平方式一样。 λ
- HorizontalIndent: 表示子组件水平偏移多少像素。 λ

此属性和“horizontalAlignment = GridData.BEGINNING”属性一起使用。下面代码设置“Button5”水平偏移 4 像素，如图 3 所示。

```
GridData gridData = new GridData();
```

```
gridData.horizontalIndent = 4;
```

```
button5.setLayoutData(gridData);
```

HorizontalSpan: 表示组件水平占据几个网格。 λ

此属性非常有用，当要设置一个组件占据几个单元格时，需要设置 HorizontalSpan 属性。例如，下面代码设置“Button5”按钮水平占据两个网格，如图 4 所示。

```
GridData gridData = new GridData();
```

```
gridData.horizontalAlignment = GridData.FILL;
```

```
gridData.horizontalSpan = 2;
```

```
button5.setLayoutData(gridData);
```

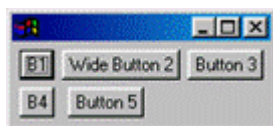


图 3 组件水平偏移

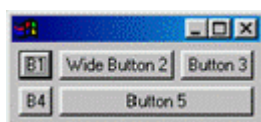


图 4 水平占据网格

●VerticalSpan: 表示组件垂直占据几个网格。

●GrabExcessHorizontalSpace: 表示当父组件大小改变时，子组件是否以水平方向抢占空间。 λ

●GrabExcessVerticalSpace: 表示当父组件大小改变时，子组件是否以垂直方向抢占空间。 λ

●WidthHint: 表示子组件的宽度为多少像素（前提是未设置其他相关属性）。

λ

●HeightHint: 表示子组件的高度为多少像素（前提是未设置其他相关属性）。

λ

另外，GridData 可以通过构造函数指定相应的属性值，有兴趣的读者可以参考 GridData 类的构造函数。

GridLayout 布局实例

为了更深入地理解 GridLayout 布局，下面通过具体的实例演示如何构建一个比较复杂的布局窗口。通过本例的学习，读者可以比较好地掌握 GridLayout 布局，代码如例程 3 所示。

例程 3 ComplexGridLayoutExample.java

```
public class ComplexGridLayoutExample {  
    static Display display;  
    static Shell shell;  
    static Text dogName;  
    static Combo dogBreed;  
    static Canvas dogPhoto;  
    static Image dogImage;  
    static List categories;  
    static Text ownerName;  
    static Text ownerPhone;  
    public static void main(String[] args) {
```

```
display = new Display();
```

```
shell = new Shell(display);
```

```
shell.setText("Dog Show Entry");
```

```
//新建 GridLayout 布局
```

```
GridLayout gridLayout = new GridLayout();
```

```
//把子组件分成 3 列显示
```

```
gridLayout.numColumns = 3;
```

```
shell.setLayout(gridLayout);
```

```
new Label(shell, SWT.NONE).setText("Dog's Name:");
```

```
dogName = new Text(shell, SWT.SINGLE | SWT.BORDER);
```

```
//新建水平填充的 GridData
```

```
GridData gridData = new GridData(GridData.HORIZONTAL_ALIGN_FILL);
```

```
//GridData 的组件占两列显示
```

```
gridData.horizontalSpan = 2;
```

```
dogName.setLayoutData(gridData);
```

```
new Label(shell, SWT.NONE).setText("Breed:");
```

```
dogBreed = new Combo(shell, SWT.NONE);
```

```
dogBreed.setItems(new String [] {"Collie", "Pitbull", "Poodle", "Scottie"});
```

```
dogBreed.setLayoutData(new GridData(GridData.HORIZONTAL_ALIGN_FILL));
```

```
Label label = new Label(shell, SWT.NONE);
```

```
label.setText("Categories");
```

```
label.setLayoutData(new GridData(GridData.HORIZONTAL_ALIGN_CENTER));
```

```
new Label(shell, SWT.NONE).setText("Photo:");
```

```
dogPhoto = new Canvas(shell, SWT.BORDER);
```

```
//gridData 两端填充
```

```
gridData = new GridData(GridData.FILL_BOTH);
```

```
//gridData 最佳宽度和高度
```

```
gridData.widthHint = 80;
```

```
gridData.heightHint = 80;
```

```
//设置 gridData 的子组件垂直占 3 行
```

```
gridData.verticalSpan = 3;
```

```
dogPhoto.setLayoutData(gridData);
```

```
//添加画布的重画事件
```

```
dogPhoto.addPaintListener(new PaintListener() {
```

```
public void paintControl(final PaintEvent event) {
```

```
if (dogImage != null) {
```

```
event.gc.drawImage(dogImage, 0, 0);
```

```
}
```

```
}
```

```
});
```

```
categories = new List(shell, SWT.MULTI | SWT.BORDER | SWT.V_SCROLL);
```

```
categories.setItems(new String [] {
```

```
"Best of Breed", "Prettiest Female", "Handsome Male",
```

```
"Best Dressed", "Fluffiest Ears", "Most Colors",
```

```
"Best Performer", "Loudest Bark", "Best Behaved",
```

```
"Prettiest Eyes", "Most Hair", "Longest Tail",
```

```
"Cutest Trick"});
```

```
gridData = new GridData(GridData.HORIZONTAL_ALIGN_FILL | GridData.
```

```
VERTICAL_ALIGN_FILL);
```

```
gridData.verticalSpan = 4;
```

```
int listHeight = categories.getItemHeight() * 12;
```

```
Rectangle trim = categories.computeTrim(0, 0, 0, listHeight);
```

```
gridData.heightHint = trim.height;
```

```
categories.setLayoutData(gridData);
```

```
Button browse = new Button(shell, SWT.PUSH);
```

```
browse.setText("Browse...");
```

```
gridData = new GridData(GridData.HORIZONTAL_ALIGN_FILL);
```

```
gridData.horizontalIndent = 5;
```

```
browse.setLayoutData(gridData);
```

```
//添加按钮的选择事件
```

```
browse.addSelectionListener(new SelectionAdapter() {
```

```
public void widgetSelected(SelectionEvent event) {
```

```
String fileName = new FileDialog(shell).open();
```

```
if (fileName != null) {
```

```
dogImage = new Image(display, fileName);
```

```
dogPhoto.redraw();
```

```
}
```

```
}
```

```
});
```

```
Button delete = new Button(shell, SWT.PUSH);
```

```
delete.setText("Delete");
```

```
gridData = new GridData(GridData.HORIZONTAL_ALIGN_FILL | GridData.
```

```
VERTICAL_ALIGN_BEGINNING);
```

```
gridData.horizontalIndent = 5;
```

```
delete.setLayoutData(gridData);
```

```
delete.addSelectionListener(new SelectionAdapter() {
```

```
public void widgetSelected(SelectionEvent event) {
```

```
if (dogImage != null) {
```

```
dogImage.dispose();
```

```
dogImage = null;
```

```
dogPhoto.redraw();
```

```
}
```

```
}
```

```
});
```

```
Group ownerInfo = new Group(shell, SWT.NONE);
```

```
ownerInfo.setText("Owner Info");
```



```
gridLayout = new GridLayout();
```

```
gridLayout.numColumns = 2;
```

```
ownerInfo.setLayout(gridLayout);
```

```
gridData = new GridData(GridData.HORIZONTAL_ALIGN_FILL);
```

```
gridData.horizontalSpan = 2;
```

```
ownerInfo.setLayoutData(gridData);
```

```
new Label(ownerInfo, SWT.NONE).setText("Name:");
```

```
ownerName = new Text(ownerInfo, SWT.SINGLE | SWT.BORDER);
```

```
ownerName.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
```

```
new Label(ownerInfo, SWT.NONE).setText("Phone:");
```

```
ownerPhone = new Text(ownerInfo, SWT.SINGLE | SWT.BORDER);
```

```
ownerPhone.setLayoutData(new GridData(GridData.FILL_HORIZONTAL));
```

```
Button enter = new Button(shell, SWT.PUSH);
```

```
enter.setText("Enter");
```

```
gridData = new GridData(GridData.HORIZONTAL_ALIGN_END);
```

```
gridData.horizontalSpan = 3;
```

```
enter.setLayoutData(gridData);
```

```
enter.addSelectionListener(new SelectionAdapter() {
```

```
public void widgetSelected(SelectionEvent event) {
```

```
System.out.println("Dog Name: " + dogName.getText());
```

```
System.out.println("Dog Breed: " + dogBreed.getText());
```

```
System.out.println("Owner Name: " + ownerName.getText());
```

```
System.out.println("Owner Phone: " + ownerPhone.getText());
```

```
System.out.println("Categories:");
```

```
String cats[] = categories.getSelection();
```

```
for (int i = 0; i < cats.length; i++) {
```

```
System.out.println("t" + cats[i]);
```

```
}
```

```
}
```

```
});
```

```
shell.pack();
```

```
shell.open();
```

```
while (!shell.isDisposed()) {
```

```
if (!display.readAndDispatch()) display.sleep();
```

```
}
```

```
if (dogImage != null) {
```

```
dogImage.dispose();
```

```
}
```

```
}
```

```
}
```

程序中构建了一个比较复杂的窗口，设置了 GridLayout 的布局信息和相关的 GridData 网格数据信息，程序运行效果如图 5 所示。

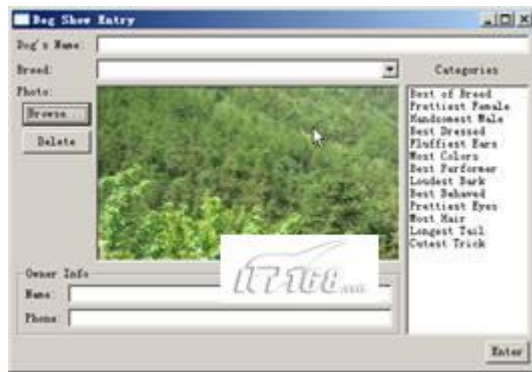


图 5 GridLayout 布局实例

注意：每个 GridData 都必须有单独的实例，几个子组件不能够共用同一个 GridData（即使几个组件有相同属性的 GridData 也必须新建几个实例）。

自定义布局

在 SWT 中，用户可以通过 setLayout 设置组件的布局信息。布局对象会根据父组件的大小和子组件的布局信息计算出每个子组件的位置和大小，使整个布局空间符合用户的需求。下面将介绍如何创建自己的布局类，实现用户自定义的布局。

Layout 类

在 SWT 中，所有的布局类都继承于 Layout 抽象类。Layout 有两个抽象方法。

1. computeSize (Composite composite, int wHint, int hHint, boolean flushCache)
2. layout (Composite composite, boolean flushCache)

computeSize 方法负责计算组件所有子组件所占的高度和宽度，并返回一个 Point 类型的变量 (width, height)。layout 方法负责计算子组件的大小和位置，并按计算出来的位置排列子组件。

创建自己的布局类

如果用户希望组件按自己的方式进行布局，可以创建自己的布局类，实现自己的布局。要实现自己的布局，用户要继承 Layout 类，并实现 layout 方法和 computeSize 方法。下面将实现一个简单的按列进行布局的布局类，在此布局中，所有的子组件将按一列显示，并且子组件的宽度相等，代码如例程 4 所示。

例程 4 ColumnLayout.java

```
public class ColumnLayout extends Layout {  
  
    public static final int MARGIN = 4;  
  
    public static final int SPACING = 2;  
  
    Point [] sizes;  
  
    int maxWidth, totalHeight;  
  
    protected Point computeSize(Composite composite, int wHint, int hHint,  
    boolean flushCache) {  
  
        Control children[] = composite.getChildren();  
  
        if (flushCache || sizes == null || sizes.length != children.length)  
        {  
            initialize(children);  
        }  
  
        int width = wHint, height = hHint;  
  
        if (wHint == SWT.DEFAULT) width = maxWidth;  
  
        if (hHint == SWT.DEFAULT) height = totalHeight;  
  
        return new Point(width + 2 * MARGIN, height + 2 * MARGIN);  
    }  
}
```

```
}
```

```
protected void layout(Composite composite, boolean flushCache) {
```

```
Control children[] = composite.getChildren();
```

```
if (flushCache || sizes == null || sizes.length != children.length)
```

```
{
```

```
initialize(children);
```

```
}
```

```
Rectangle rect = composite.getClientArea();
```

```
int x = MARGIN, y = MARGIN;
```

```
//计算最大宽度
```

```
int width = Math.max(rect.width - 2 * MARGIN, maxWidth);
```

```
for (int i = 0; i < children.length; i++) {
```

```
int height = sizes[i].y;
```

```
//设置子组件的位置
```

```
children[i].setBounds(x, y, width, height);
```

```
//计算当前组件的 y 轴的坐标
```

```
y += height + SPACING;
```

```
}
```

```
}
```

```
void initialize(Control children[]) {
```

```
maxWidth = 0;
```

```
totalHeight = 0;
```

```

sizes = new Point [children.length];

for (int i = 0; i < children.length; i++) {

    //计算子组件的大小

    sizes[i] = children[i].computeSize(SWT.DEFAULT, SWT.DEFAULT, true);

    maxWidth = Math.max(maxWidth, sizes[i].x);

    totalHeight += sizes[i].y;

}

totalHeight += (children.length - 1) * SPACING;

}

}

```

在 ColumnLayout 类中，通过 layout 方法对子组件重新计算位置，并设置子组件的位置。为了验证 ColumnLayout 类，下面通过 ColumnLayoutTest 类测试 ColumnLayout 类的效果，代码如例程 5 所示。

例程 5 ColumnLayoutTest.java

```

public class ColumnLayoutTest {

    static Shell shell;

    static Button button3;

    public static void main(String[] args) {

        Display display = new Display();

        shell = new Shell(display);

        shell.setLayout(new ColumnLayout());

        new Button(shell, SWT.PUSH).setText("B1");
    }
}

```

```
new Button(shell, SWT.PUSH).setText("Very Wide Button 2");
```

```
(button3 = new Button(shell, SWT.PUSH)).setText("Button 3");
```

```
new Text(shell, SWT.NONE).setText("text");
```

```
Button grow = new Button(shell, SWT.PUSH);
```

```
grow.setText("Grow Button 3");
```

```
// 添加选择组件监听器
```

```
grow.addSelectionListener(new SelectionAdapter() {
```

```
public void widgetSelected(SelectionEvent e) {
```

```
button3.setText("Extreemely Wide Button 3");
```

```
//组件大小改变后通知父组件进行重新布局
```

```
shell.layout();
```

```
shell.pack();
```

```
}
```

```
});
```

```
Button shrink = new Button(shell, SWT.PUSH);
```

```
shrink.setText("Shrink Button 3");
```

```
shrink.addSelectionListener(new SelectionAdapter() {
```

```
public void widgetSelected(SelectionEvent e) {
```

```
button3.setText("Button 3");
```

```
//组件大小改变后通知父组件进行重新布局
```

```
shell.layout();
```

```
shell.pack();
```

```

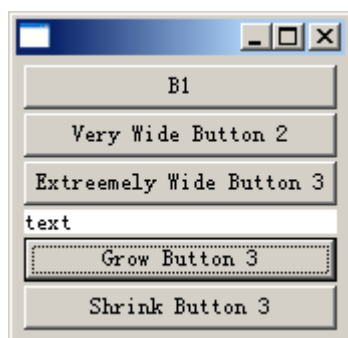
}
});
shell.pack();
shell.open();
while (!shell.isDisposed()) {
    if (!display.readAndDispatch()) display.sleep();
}
}
}
}

```

当选择“Grow Button 3”组件后，layout 方法会根据子组件的最大宽度调整所有子组件的宽度，程序运行效果如图 6 所示。



原始大小



宽度改变后

图 6 自己定义布局

本节通过实例介绍了几种常用的布局方式，读者可以通过这几种布局方式实现 SWT 中大多数的布局需求。另外还有一种常用的布局 `FormLayout`，有兴趣读者可以自行研究，在这里不一一介绍。

如果有某些比较特殊的要求，读者可以尝试修改布局类，以适应相关的布局。读者应该掌握如何设置组件相应的布局信息，掌握如何使用几种方式进行布局，特别是 `GridLayout` 布局方式。在有特殊需要的时候要能够修改布局类以适应自己的要求。