[FormLayout 1](#_Toc322616282)

[FormLayout Configuration Fields 1](#_Toc322616283)

[FormData Object Fields 1](#_Toc322616284)

[FormAttachment Objects 3](#_Toc322616285)

[Attaching to a Position 3](#_Toc322616286)

[Attaching to Another Widget 5](#_Toc322616287)

[A FormLayout Example 7](#_Toc322616288)

[Full example code 9](#_Toc322616289)

## FormLayout

FormLayout works by creating FormAttachments for each side of the widget, and storing them in the layout data. An attachment 'attaches' a specific side of the widget either to a position in the parent Composite or to another widget within the layout. This provides tremendous flexibility when laying out, as it allows you to specify the placement of individual widgets within the layout.

### FormLayout Configuration Fields

The marginWidth, and MarginHeight fields in FormLayout are similar to those in GridLayout. Left and right margins are defined by marginWidth, and top and bottom margins are defined by marginHeight. Margins can also be defined on a per-widget basis in the attachments. FormLayout margins are zero by default.

To set the margins, we create a FormLayout, and set the margin fields. The following code will set a margin of five pixels around all four sides of the parent Composite:

Display display = new Display ();

Shell shell = new Shell (display);

FormLayout layout= new FormLayout ();

layout.marginHeight = 5;

layout.marginWidth = 5;

shell.setLayout(layout);

### FormData Object Fields

FormData objects specify how each widget in a FormLayout will be laid out. Each FormData object defines the attachments for all four sides of the widget. These attachments tell where to position each side of the widget. To set a widget's FormData object, you use the setLayoutData(Object) method, for example:

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

button1.setText("B1");

button1.setLayoutData(new FormData());

This code creates a FormData object with no attachments. In this case, default attachments are defined, which defeats the whole purpose and utility of FormLayout. The default attachments attach the widget to the top and left edges of the parent Composite. If every widget in a FormLayout used the default attachments, they would all be laid out one on top of another in the top left corner of the parent Composite.

The left, right, top, and bottom fields of FormData specify the FormAttachment objects that are associated with the left, right, top and bottom sides of the widget, respectively. These fields are set in the following example:

FormData formData = new FormData();

formData.top = new FormAttachment(0,60);

formData.bottom = new FormAttachment(100,-5);

formData.left = new FormAttachment(20,0);

formData.right = new FormAttachment(100,-3);

button1.setLayoutData(formData);

http://www.eclipse.org/articles/Article-Understanding-Layouts/images/FormLayoutSample01.jpg

A FormAttachment object defines the attachment of a specific side of a widget. There are many ways that a side can be attached: to a position in the parent Composite, to an edge of the Composite, to the adjacent side of another widget, to the opposite side of another widget, or centered on another widget. Attaching to a position places the side of the widget so that it is always at a percentage of the Composite. To attach to an edge of the Composite, the percentage is either 0% or 100%. Attaching to the adjacent side of another widget ensures that the specified side of the widget is always next to the closest side of the other widget. Attaching to the opposite side of another widget ensures that the specific side of the widget is aligned with the furthest side of the other widget. Finally, attaching to the center of another widget centers the widget on the other widget. Any of these ways can be done with or without an offset.

The width and height fields of FormData specify the requested width and the height of the widget. If a requested width or height conflicts with constraints set by the attachments, then that width or height will not be honored. Although setting attachments can also determine width and height, there are some cases when you do not want to define attachments for all sides of the widget. In this case, it may be useful to set the width and height of the widget as follows:

FormData formData = new FormData(20,30);

formData.top = new FormAttachment(0,60);

formData.left = new FormAttachment(20,0);

button1.setLayoutData(formData);

If you wish to set only the width or the height, you can directly set the width or height field in the FormData object:

FormData formData = new FormData();

formData.width = 30;

formData.top = new FormAttachment(0,60);

formData.bottom = new FormAttachment(100,-5);

formData.left = new FormAttachment(20,0);

button1.setLayoutData(formData);

Note that if a button is attached to the parent Composite on both sides, when the Composite is resized, the button will grow or shrink along with it.

### FormAttachment Objects

A FormAttachment is an object that defines the attachment for a specific side of a widget. It is not always necessary to set an attachment for all four sides of a widget. Often, specifying one or more sides of a widget can fully specify its placement in the layout. In order to properly place your widgets, you should define an attachment for at least one of left or right in the FormData, and at least one of top or bottom. If you only wish to attach the left side of a widget and not the right, then the widget will be positioned based on its left side, and the widget will take its natural size (or its requested size, if one was set for it). If you do not attach the left or the right, default positioning will attach your widget to the left side of the form. The same logic applies for the top and bottom sides.

### Attaching to a Position

There are many types of attachment. The first is to attach the widget to a position in the parent Composite. This can be done by defining a percentage value out of 100, for example:

FormData formData = new FormData();

formData.top = new FormAttachment(50,0);

button1.setLayoutData(formData);

http://www.eclipse.org/articles/Article-Understanding-Layouts/images/FormLayoutSamplePosition50.jpg

This sets the top of the Button to a position that represents 50% of the height of the parent Composite (a Shell), with an offset of 0. When the shell is resized, the top side of theButton will still be at 50%, like so:



If we chose to set an offset value, the top side of the Button would have been set to 50% of the Composite plus or minus the number of pixels set for the offset.

We can also define the position of the button using an arbitrary scale, for example:

FormData formData = new FormData();

formData.top = new FormAttachment(30,70,10);

button1.setLayoutData(formData);

If the height of the Composite is defined as being 70 units, this sets the top of the Button to a position representing 30 units down from the top of the Composite, plus 10 pixels (i.e. 3/7ths of the height of the composite plus 10 pixels).

To attach a side of a widget to an edge of the parent Composite, set the position to either 0% or 100%. The 0 position is defined as the top of the Composite when going vertically, and the left when going horizontally. The right and bottom edges of the Composite are defined as the 100 position. Therefore, if we want to attach a widget to the right edge of the Composite, we simply have to create an attachment that sets the position to 100:

FormData formData = new FormData();

formData.right = new FormAttachment(100,-5);

button1.setLayoutData(formData);

http://www.eclipse.org/articles/Article-Understanding-Layouts/images/FormLayoutSampleParent100.jpg

This attaches the right side of the Button to the right edge of the parent (a Shell), with an offset of five pixels. Note that the offsets go in one direction only. If you want a widget offset down or to the right, the offset should be positive. For offsets that shift the widget up or to the left, the offset should be negative. When the Shell is resized, the Buttonwill always be five pixels away from the right edge:



### Attaching to Another Widget

The third type of attachment is to attach the side of the widget to another control within the parent Composite. The side can be attached to the adjacent side of the other control (the default), to the opposite side of the other control, or the widget can be centered on the other control, all with or without and offset.

The most common way to attach to another control is to attach to its adjacent side. For example, the following code:

FormData formData = new FormData();

formData.top = new FormAttachment(20,0);

button1.setLayoutData(formData);

FormData formData2 = new FormData();

formData2.top = new FormAttachment(button1,10);

button2.setLayoutData(formData2);



This example attaches the top of button2 to the bottom of button1. Note that when the window is resized, button1 will move so that its top side is always positioned at 20% of the Shell, and button2 will move so that its top side is always 10 pixels below the adjacent (bottom) side of button1.



While the default is to attach the side of a widget to the adjacent side of a control, FormAttachments can also be created to attach to the opposite side of a control. This is useful when lining up widgets. In this case, you create the attachment to the other control using TOP, BOTTOM, LEFT or RIGHT alignment, for example:

formData2.top = new FormAttachment(button1,0,SWT.TOP);

In the following example, the top side of button1 is positioned at 20% of the Shell. button2's top side is aligned with button1's top side, using TOP alignment. This means that the top side of button2 is also positioned at 20% of the Shell. Note that when specifying the top attachment, only the vertical placement of the widget is being defined. It is still necessary to set the left attachment for button2 so that the Buttons are not stacked on top of each other.

FormData formData = new FormData(50,50);

formData.top = new FormAttachment(20,0);

button1.setLayoutData(formData);

FormData formData2 = new FormData();

formData2.left = new FormAttachment(button1,5);

formData2.top = new FormAttachment(button1,0,SWT.TOP);

button2.setLayoutData(formData2);



The final way to attach a widget to another control is to center it on the other control. This is useful when the widgets are different sizes. In this case, you create the attachment to the other control with CENTER alignment, for example:

formData.top = new FormAttachment(button1,0,SWT.CENTER);

This will place the top of the widget in a position that will allow the widget to be centered on the other control, with an offset of 0. Setting only the top, or the bottom, or both as a center attachment will produce the same result. The top side of the widget is not centered, but the entire widget is centered, so this only needs to be specified once. Here is an example:

FormData formData1 = new FormData (50,50);

button1.setLayoutData(formData1);

FormData formData2 = new FormData ();

formData2.left = new FormAttachment (button1,5);

formData2.top = new FormAttachment (button1,0,SWT.CENTER);

button2.setLayoutData(formData2);

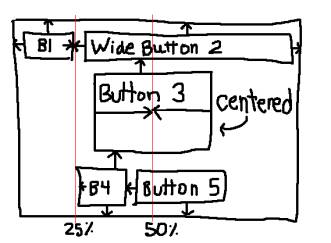


Using the different types of FormAttachment allows layouts to be defined in many different ways. FormLayout covers certain cases that cannot be solved using FillLayout,RowLayout or GridLayout, making it a very useful class for defining layouts.

**Important**: Do not define circular attachments. For example, do not attach the right edge of button1 to the left edge of button2 and then attach the left edge button2 to the right edge of button1. This will over-constrain the layout, causing undefined behavior. The algorithm will terminate, but the results are undefined. Therefore, make sure that you do not over-constrain your widgets. Only provide the attachments necessary to properly lay out the widgets.

### A FormLayout Example

So far, all the examples using FormLayout have involved one or two Buttons, to show how FormAttachments work. Next, we will do a simple example using more Buttons to show how a layout can be arranged using the attachments. We'll start by drawing a basic diagram outlining the attachments that we wish to create.



FormData data1 = new FormData();

data1.left = new FormAttachment(0,5);

data1.right = new FormAttachment(25,0);

button1.setLayoutData(data1);

FormData data2 = new FormData();

data2.left = new FormAttachment(button1,5);

data2.right = new FormAttachment(100,-5);

button2.setLayoutData(data2);

FormData data3 = new FormData(60,60);

data3.top = new FormAttachment(button1,5);

data3.left = new FormAttachment(50,-30);

data3.right = new FormAttachment(50,30);

button3.setLayoutData(data3);

FormData data4 = new FormData();

data4.top = new FormAttachment(button3,5);

data4.bottom = new FormAttachment(100,-5);

data4.left = new FormAttachment(25,0);

button4.setLayoutData(data4);

FormData data5 = new FormData();

data5.bottom = new FormAttachment(100,-5);

data5.left = new FormAttachment(button4,5);

button5.setLayoutData(data5);

In this case, since no top attachment was defined for button1 or button2, they are attached to the top of the layout. button3 is centred in the layout using percentages and offsets on the left and right sides. button4 and button5 are attached to the bottom of the layout with a five pixel offset.



When we resize, the attachments become more visible. button1 is attached on the left and the right side, so when the window is resized, it grows. Note that the right side will always be at 25% of the window. The same resize results apply for button2, as both sides are attached. The left side is attached to button1, so it will always be at 25% plus five pixels. button3 stays in the center of the window, horizontally. button4 is attached at the top and the bottom, so it grows vertically when the window is resized, but it is only attached on the left and not the right, so it does not grow horizontally. button5 will not grow or shrink, but it will always stay five pixels away from button4 on the left, and five pixels away from the bottom of the window.

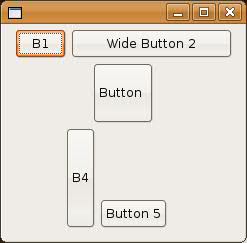


Figure . A Complex FormLayout Example

To illustrate how FormLayout can be used for more complicated arrangements, the Dog Show Entry example done previously for GridLayout is redone using FormLayout. This code produces an identical layout, but uses different concepts to achieve it.

#### Full example code

import org.eclipse.swt.SWT;

import org.eclipse.swt.events.PaintEvent;

import org.eclipse.swt.events.PaintListener;

import org.eclipse.swt.events.SelectionAdapter;

import org.eclipse.swt.events.SelectionEvent;

import org.eclipse.swt.graphics.Image;

import org.eclipse.swt.layout.FormAttachment;

import org.eclipse.swt.layout.FormData;

import org.eclipse.swt.layout.FormLayout;

import org.eclipse.swt.widgets.Button;

import org.eclipse.swt.widgets.Canvas;

import org.eclipse.swt.widgets.Combo;

import org.eclipse.swt.widgets.Display;

import org.eclipse.swt.widgets.FileDialog;

import org.eclipse.swt.widgets.Group;

import org.eclipse.swt.widgets.Label;

import org.eclipse.swt.widgets.List;

import org.eclipse.swt.widgets.Shell;

import org.eclipse.swt.widgets.Text;

public class DogShowRegistrationWindowWithFormLayout {

Image dogImage;

Text dogNameText;

Combo dogBreedCombo;

Canvas dogPhoto;

List categories;

Text nameText;

Text phoneText;

public static void main(String[] args) {

Display display = new Display();

Shell shell = new DogShowRegistrationWindow().createShell(display);

shell.open();

while (!shell.isDisposed()) {

if (!display.readAndDispatch())

display.sleep();

}

}

public Shell createShell(final Display display) {

final Shell shell = new Shell(display);

FormLayout layout = new FormLayout();

layout.marginWidth = 5;

layout.marginHeight = 5;

shell.setLayout(layout);

shell.setText("Dog Show Entry");

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

ownerInfo.setText("Owner Info");

FormLayout ownerLayout = new FormLayout();

ownerLayout.marginWidth = 5;

ownerLayout.marginHeight = 5;

ownerInfo.setLayout(ownerLayout);

Label dogName = new Label(shell, SWT.NONE);

dogName.setText("Dog's Name:");

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

Label dogBreed = new Label(shell, SWT.NONE);

dogBreed.setText("Breed:");

dogBreedCombo = new Combo(shell, SWT.NONE);

dogBreedCombo.setItems(new String[] { "Collie", "Pitbull", "Poodle",

"Scottie", "Black Lab" });

Label photo = new Label(shell, SWT.NONE);

photo.setText("Photo:");

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

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

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

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

delete.setText("Delete");

Label cats = new Label(shell, SWT.NONE);

cats.setText("Categories");

categories = new List(shell, SWT.MULTI | SWT.BORDER | SWT.V\_SCROLL

| SWT.H\_SCROLL);

categories.setItems(new String[] { "Best of Breed", "Prettiest Female",

"Handsomest Male", "Best Dressed", "Fluffiest Ears",

"Most Colors", "Best Performer", "Loudest Bark",

"Best Behaved", "Prettiest Eyes", "Most Hair", "Longest Tail",

"Cutest Trick" });

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

enter.setText("Enter");

FormData data = new FormData();

data.top = new FormAttachment(dogNameText, 0, SWT.CENTER);

dogName.setLayoutData(data);

data = new FormData();

data.left = new FormAttachment(dogName, 5);

data.right = new FormAttachment(100, 0);

dogNameText.setLayoutData(data);

data = new FormData();

data.top = new FormAttachment(dogBreedCombo, 0, SWT.CENTER);

dogBreed.setLayoutData(data);

data = new FormData();

data.top = new FormAttachment(dogNameText, 5);

data.left = new FormAttachment(dogNameText, 0, SWT.LEFT);

data.right = new FormAttachment(categories, -5);

dogBreedCombo.setLayoutData(data);

data = new FormData(80, 80);

data.top = new FormAttachment(dogBreedCombo, 5);

data.left = new FormAttachment(dogNameText, 0, SWT.LEFT);

data.right = new FormAttachment(categories, -5);

data.bottom = new FormAttachment(ownerInfo, -5);

dogPhoto.setLayoutData(data);

dogPhoto.addPaintListener(new PaintListener() {

public void paintControl(final PaintEvent event) {

if (dogImage != null) {

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

}

}

});

data = new FormData();

data.top = new FormAttachment(dogPhoto, 0, SWT.TOP);

photo.setLayoutData(data);

data = new FormData();

data.top = new FormAttachment(photo, 5);

data.right = new FormAttachment(dogPhoto, -5);

browse.setLayoutData(data);

browse.addSelectionListener(new SelectionAdapter() {

public void widgetSelected(SelectionEvent event) {

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

if (fileName != null) {

dogImage = new Image(display, fileName);

}

}

});

data = new FormData();

data.left = new FormAttachment(browse, 0, SWT.LEFT);

data.top = new FormAttachment(browse, 5);

data.right = new FormAttachment(dogPhoto, -5);

delete.setLayoutData(data);

delete.addSelectionListener(new SelectionAdapter() {

public void widgetSelected(SelectionEvent event) {

if (dogImage != null) {

dogImage.dispose();

dogImage = null;

dogPhoto.redraw();

}

}

});

data = new FormData(90, 140);

data.top = new FormAttachment(dogPhoto, 0, SWT.TOP);

data.right = new FormAttachment(100, 0);

data.bottom = new FormAttachment(enter, -5);

categories.setLayoutData(data);

data = new FormData();

data.bottom = new FormAttachment(categories, -5);

data.left = new FormAttachment(categories, 0, SWT.CENTER);

cats.setLayoutData(data);

data = new FormData();

data.right = new FormAttachment(100, 0);

data.bottom = new FormAttachment(100, 0);

enter.setLayoutData(data);

enter.addSelectionListener(new SelectionAdapter() {

public void widgetSelected(SelectionEvent event) {

System.out.println("\nDog Name: " + dogNameText.getText());

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

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

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

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

String cats[] = categories.getSelection();

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

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

}

}

});

data = new FormData();

data.bottom = new FormAttachment(enter, -5);

data.left = new FormAttachment(0, 0);

data.right = new FormAttachment(categories, -5);

ownerInfo.setLayoutData(data);

Label name = new Label(ownerInfo, SWT.NULL);

name.setText("Name:");

Label phone = new Label(ownerInfo, SWT.PUSH);

phone.setText("Phone:");

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

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

data = new FormData();

data.top = new FormAttachment(nameText, 0, SWT.CENTER);

name.setLayoutData(data);

data = new FormData();

data.top = new FormAttachment(phoneText, 0, SWT.CENTER);

phone.setLayoutData(data);

data = new FormData();

data.left = new FormAttachment(phone, 5);

data.right = new FormAttachment(100, 0);

nameText.setLayoutData(data);

data = new FormData();

data.left = new FormAttachment(nameText, 0, SWT.LEFT);

data.right = new FormAttachment(100, 0);

data.top = new FormAttachment(55, 0);

phoneText.setLayoutData(data);

shell.pack();

return shell;

}

}

This is what the layout looks like after Mary Smith enters Bifford in the dog show:



When the window is resized, the same controls are resized as in the GridLayout example.

