

# SWT

---

## *A Developer's Notebook™*

Tim Hatton

- Composite Objects
- Tree Widgets
- Table Widgets
- Menu Systems
- Standard Dialogs

O'REILLY®

# SWT CoolBars

Chapter 4 demonstrated the creation and use of SWT ToolBars. This chapter examines another set of classes that can be used to create ToolBar-like constructs—the SWT CoolBar and CoolItem classes from the `org.eclipse.swt.widgets` package.

A CoolBar is like a ToolBar on steroids. Like ToolBar and ToolItems, CoolBar is a container to which you add instances of the CoolItem class. Although a CoolItem object appears as a button on the ToolBar, a CoolItem serves only as a container for other widget types—Combo, Text, or Button. This makes a CoolBar a massively useful interface construct. In fact, the Eclipse toolbar is actually a CoolBar and is a great example of how to leverage the capability of the CoolBar and CoolItem classes to enable the user to customize the user interface.

## Creating a CoolBar

The steps required to create a CoolBar are similar to those required to create a ToolBar, except that an instance of CoolBar is created instead of ToolBar and instances of CoolItem are created in place of the ToolItem instances for the individual buttons. The similarity ends there, however.

The CoolItem objects created serve only as containers for other widgets. If you want to create a CoolBar similar in appearance to the ToolBar from Chapter 4, you must create individual Button objects that are added to the corresponding CoolItem. A good understanding of CoolBar can be gained by going through the process of creating a CoolBar similar in appearance to the ToolBar created in Chapter 4.

### *In this chapter:*

- *Creating a CoolBar*
- *Using Events with CoolBars*
- *Adding Widgets Other Than Buttons to the CoolBar*
- *Preventing the User from Rearranging the CoolBar*
- *Using ToolBars with CoolBars*

## How do I do that?

1. Create an instance of the CoolBar class and set its initial position and size.
2. Create an instance of the CoolItem class for each widget you wish to appear on the CoolBar.
3. Create a widget and assign it to the corresponding CoolItem.

Consider Example 15-1, which creates a CoolBar similar to the ToolBar created in Chapter 4.

### Example 15-1. Creating a CoolBar

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.events.HelpEvent;
import org.eclipse.swt.events.HelpListener;
import org.eclipse.swt.graphics.Image;
import org.eclipse.swt.graphics.Point;
import org.eclipse.swt.widgets.*;

public class CoolbarShellExample {
    Display d;
    Shell s;

    CoolbarShellExample() {
        d = new Display();
        s = new Shell(d);
        s.setSize(300,300);
        s.setImage(new Image(d, "c:\\icons\\JavaCup.ico"));
        s.setText("A Shell Coolbar Example");

        final CoolBar bar = new CoolBar(s,SWT.BORDER);
        bar.setSize(280,70);
        bar.setLocation(0,0);
        // create images for coolbar buttons
        final Image saveIcon = new Image(d, "c:\\icons\\save.jpg");
        final Image openIcon = new Image(d, "c:\\icons\\open.jpg");
        final Image cutIcon = new Image(d, "c:\\icons\\cut.jpg");
        final Image copyIcon = new Image(d, "c:\\icons\\copy.jpg");
        final Image pasteIcon = new Image(d, "c:\\icons\\paste.jpg");

        // create and add the button for performing an open operation
        final CoolItem openCoolItem = new CoolItem(bar, SWT.NONE);
        final Button openBtn = new Button(bar, SWT.PUSH);
        openBtn.setImage(openIcon);
        openBtn.pack();
        Point size = openBtn.getSize();
        openCoolItem.setControl(openBtn);
        openCoolItem.setSize(openCoolItem.computeSize(size.x, size.y));

        //create and add the button for performing a save operation
        final CoolItem saveCoolItem = new CoolItem(bar, SWT.PUSH);
```

**Example 15-1.** Creating a CoolBar (continued)

```
final Button saveBtn = new Button(bar, SWT.PUSH);
saveBtn.setImage(saveIcon);
saveBtn.pack();
size = saveBtn.getSize();
saveCoolItem.setControl(saveBtn);
saveCoolItem.setSize(saveCoolItem.computeSize(size.x, size.y));

//create and add the button for performing a cut operation
final CoolItem cutCoolItem = new CoolItem(bar, SWT.PUSH);
final Button cutBtn = new Button(bar, SWT.PUSH);
cutBtn.setImage(cutIcon);
cutBtn.pack();
size = cutBtn.getSize();
cutCoolItem.setControl(cutBtn);
cutCoolItem.setSize(cutCoolItem.computeSize(size.x, size.y));

// create and add the button for performing a copy operation
final CoolItem copyCoolItem = new CoolItem(bar, SWT.PUSH);
final Button copyBtn = new Button(bar, SWT.PUSH);
copyBtn.setImage(copyIcon);
copyBtn.pack();
size = copyBtn.getSize();
copyCoolItem.setControl(copyBtn);
copyCoolItem.setSize(copyCoolItem.computeSize(size.x, size.y));

// create and add the button for performing a paste operation
final CoolItem pasteCoolItem = new CoolItem(bar, SWT.PUSH);
final Button pasteBtn = new Button(bar, SWT.PUSH);
pasteBtn.setImage(pasteIcon);
pasteBtn.pack();
size = pasteBtn.getSize();
pasteCoolItem.setControl(pasteBtn);
pasteCoolItem.setSize(pasteCoolItem.computeSize(size.x, size.y));
pasteCoolItem.setMinimumSize(size);

openBtn.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent event) {
        System.out.println("Open");
    }
});

saveBtn.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent event) {
        System.out.println("Save");
    }
});

cutBtn.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent event) {
        System.out.println("Cut");
    }
});
```

### Example 15-1. Creating a CoolBar (continued)

```
copyBtn. addSelectionListener(new SelectionAdapter() {
    public void void widgetSelected(SelectionEvent event) {
        System.out.println("Copy");
    }
});

pasteBtn. addSelectionListener(new SelectionAdapter() {
    public void void widgetSelected(SelectionEvent event) {
        System.out.println("Paste");
    }
});

// create the menu
Menu m = new Menu(s,SWT.BAR);

// create a File menu and add an Exit item
final MenuItem file = new MenuItem(m, SWT.CASCADE);
file.setText("&File");
final Menu filemenu = new Menu(s, SWT.DROP_DOWN);
file.setMenu(filemenu);
final MenuItem openMenuItem = new MenuItem(filemenu, SWT.PUSH);
openMenuItem.setText("&Open\tCTRL+O");
openMenuItem.setAccelerator(SWT.CTRL+'O');
final MenuItem saveMenuItem = new MenuItem(filemenu, SWT.PUSH);
saveMenuItem.setText("&Save\tCTRL+S");
saveMenuItem.setAccelerator(SWT.CTRL+'S');
final MenuItem separator = new MenuItem(filemenu, SWT.SEPARATOR);
final MenuItem exitMenuItem = new MenuItem(filemenu, SWT.PUSH);
exitMenuItem.setText("E&xit");

// create an Edit menu and add Cut, Copy, and Paste items
final MenuItem edit = new MenuItem(m, SWT.CASCADE);
edit.setText("&Edit");
final Menu editmenu = new Menu(s, SWT.DROP_DOWN);
edit.setMenu(editmenu);
final MenuItem cutMenuItem = new MenuItem(editmenu, SWT.PUSH);
cutMenuItem.setText("&Cut");
final MenuItem copyMenuItem = new MenuItem(editmenu, SWT.PUSH);
copyMenuItem.setText("Co&py");
final MenuItem pasteMenuItem = new MenuItem(editmenu, SWT.PUSH);
pasteMenuItem.setText("&Paste");

//create a Window menu and add Child items
final MenuItem window = new MenuItem(m, SWT.CASCADE);
window.setText("&Window");
final Menu windowmenu = new Menu(s, SWT.DROP_DOWN);
window.setMenu(windowmenu);
final MenuItem maxMenuItem = new MenuItem(windowmenu, SWT.PUSH);
maxMenuItem.setText("Ma&ximize");
final MenuItem minMenuItem = new MenuItem(windowmenu, SWT.PUSH);
minMenuItem.setText("Mi&nimize");
```

### Example 15-1. Creating a CoolBar (continued)

```
// create a Help menu and add an About item
final MenuItem help = new MenuItem(m, SWT.CASCADE);
help.setText("&Help");
final Menu helpmenu = new Menu(s, SWT.DROP_DOWN);
help.setMenu(helpmenu);
final MenuItem aboutMenuItem = new MenuItem(helpmenu, SWT.PUSH);
aboutMenuItem.setText("&About");

// add action listeners for the menu items

openMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Open");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

saveMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Save");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

exitMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.exit(0);
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

cutMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Cut");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

copyMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Copy");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

pasteMenuItem.addSelectionListener(new SelectionListener() {
```

### Example 15-1. Creating a CoolBar (continued)

```
        public void widgetSelected(SelectionEvent e) {
            System.out.println("Paste");
        }
        public void widgetDefaultSelected(SelectionEvent e) {
        }
    });

    maxMenuItem.addSelectionListener(new SelectionListener() {
        public void widgetSelected(SelectionEvent e) {
            Shell parent = (Shell)maxItem.getParent().getParent();
            parent.setMaximized(true);
        }
        public void widgetDefaultSelected(SelectionEvent e) {
        }
    });

    minMenuItem.addSelectionListener(new SelectionListener() {
        public void widgetSelected(SelectionEvent e) {
            Shell parent = (Shell)minItem.getParent().getParent();
            parent.setMaximized(false);
        }
        public void widgetDefaultSelected(SelectionEvent e) {
        }
    });

    aboutMenuItem.addSelectionListener(new SelectionListener() {
        public void widgetSelected(SelectionEvent e) {
            System.out.println("Help Invoked");
        }
        public void widgetDefaultSelected(SelectionEvent e) {
        }
    });

    s.setMenuBar(m);

    s.open();
    while(!s.isDisposed()){
        if(!d.readAndDispatch())
            d.sleep();
    }
    d.dispose();
}
```

When executed, CoolbarShellExample, shown in Example 15-1, creates Figure 15-1.

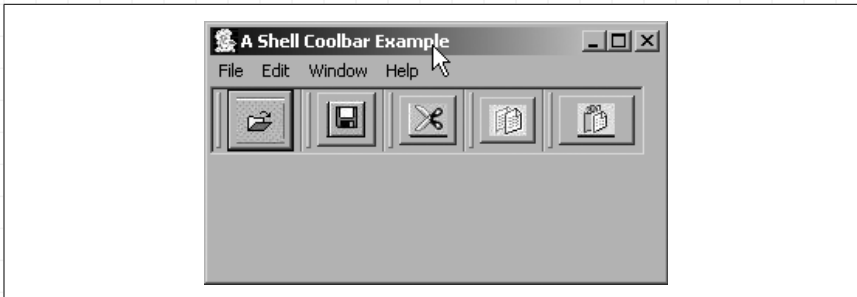


Figure 15-1. CoolBarShellExample

## What just happened?

The code that creates the CoolBar itself is straightforward. First, an instance of CoolBar is created, then sized and positioned to fit the Shell:

```
final CoolBar bar = new CoolBar(s, SWT.BORDER);
bar.setSize(280,70);
bar.setLocation(0,0);
```

You can understand the steps required to create and display a complete set of CoolBar buttons by examining of the creation of just one of the CoolItem objects:

```
final CoolItem openCoolItem = new CoolItem(bar, SWT.NONE);
final Button openBtn = new Button(bar, SWT.PUSH);
openBtn.setImage(openIcon);
openBtn.pack();
Point size = openBtn.getSize();
openCoolItem.setControl(openBtn);
openCoolItem.setSize(openCoolItem.computeSize(size.x, size.y));
```

Here, an instance of CoolItem is created. CoolItem serves as the placeholder on the CoolBar to which you attach the widget you wish to place on the CoolBar. In this example, a Button is created and added to the CoolItem using setControl(). The remaining code sets the size of the CoolItem sufficiently to display the Button. The example code repeats this process for all the Button objects on the CoolBar.

## Using Events with CoolBars

Unlike ToolItem, no events are associated directly with a CoolItem. Rather, events are associated with the widget that is assigned to the CoolItem.

## How do I do that?

Example 15-1 assigns a `SelectionListener` to each `Button` to handle the event caused by the user clicking the button:

```
openBtn.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent event) {
        System.out.println("Open");
    }
});
```

## Adding Widgets Other Than Buttons to the CoolBar

It is possible to add other types of widgets to the `CoolBar` in the same manner as was done in Example 15-1 with `Button`. However, it only makes sense to consider a couple widget types for inclusion on a `CoolBar`. `List`, for example, is not a good candidate, since the room required for the `List` would exceed the amount of space available (top to bottom) on the `CoolBar`.

*Not only that, but it would be ugly.*

`Combo` is a good candidate for inclusion on a `CoolBar` when you must enable your user to select an item from a list.

## How do I do that?

You add a `Combo` to the `CoolBar` using the same steps that are used to add a `Button`. Use this code to add a `Combo` to `CoolbarShellExample` from Example 15-1:

```
final CoolItem fontCoolItem = new CoolItem(bar, SWT.PUSH);
final Combo fontCombo = new Combo(bar, SWT.READ_ONLY | SWT.BORDER);
String[] items = {"Arial", "Courier", "Times New Roman"};
fontCombo.setItems(items);
fontCombo.pack();
size = fontCombo.getSize();
fontCoolItem.setControl(fontCombo);
fontCoolItem.setSize(fontCoolItem.computeSize(size.x, size.y));
fontCoolItem.setMinimumSize(size);
```

Doing so yields the result shown in Figure 15-2.

A `selectionListener` can be associated with the `Combo` to take appropriate action when the user selects an item.

```
fontCombo.addSelectionListener(new SelectionAdapter() {
    public void widgetSelected(SelectionEvent event) {
        System.out.println("Change Font Here");
    }
});
```

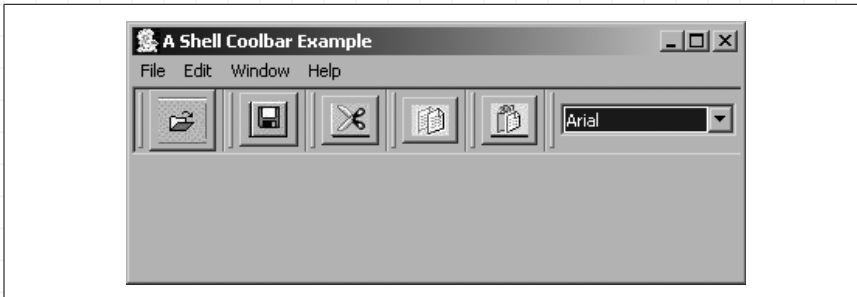


Figure 15-2. Using a Combo on a CoolBar

## Preventing the User from Rearranging the CoolBar

Executing the `CoolbarShellExample` program demonstrates how the user can interact with the CoolBar. Each Button can be dragged and dropped into a new location on the CoolBar in any manner the user sees fit. The Button objects can also be resized by the user.

There are times when this functionality may not be desirable. In those situations, you must take steps to prevent the user from rearranging or resizing the CoolItems.

### How do I do that?

Adding this line to the `CoolbarShellExample` following the code that creates the last CoolItem results in the CoolBar being locked and prevents the user from rearranging or resizing the CoolItems:

```
bar.setLocked(true);
```

## Using ToolBars with CoolBars

A close examination of `CoolbarShellExample` reveals several issues. First, Button objects associated with the CoolBar are unable to use both text and an image, as was the case with `ToolBar`. Attempting to use both text and image results in only text being visible. Second, since the user can move CoolItems around on the CoolBar, you lose the ability to maintain functional groups of buttons unless you lock the CoolBar (which negates many of the reasons for using it in the first place).

These issues are easily solved once you realize that it is possible to add a `ToolBar` to a CoolBar as a single unit.

## How do I do that?

Any `ToolBar` can be added to a `CoolBar`. All you must do is create the `ToolBar` in the normal manner (see Chapter 4), then create a `CoolItem` to serve as the holder for the `ToolBar`. Consider the modification to `CoolbarShellExample` shown in Example 15-2.

### Example 15-2. Using a `ToolBar` with a `CoolBar`

```
import org.eclipse.swt.SWT;
import org.eclipse.swt.graphics.Image;
import org.eclipse.swt.graphics.Point;
import org.eclipse.swt.widgets.*;

public class CoolbarShellExample {
    Display d;
    Shell s;

    CoolbarShellExample() {
        d = new Display();
        s = new Shell(d);
        s.setSize(400,300);
        s.setImage(new Image(d, "c:\\icons\\JavaCup.ico"));
        s.setText("A Shell Coolbar Example");

        final CoolBar coolBar = new CoolBar(s,SWT.BORDER);
        coolBar.setSize(395,70);
        coolBar.setLocation(0,0);
        // create images for toolbar buttons
        final Image saveIcon = new Image(d, "c:\\icons\\save.jpg");
        final Image openIcon = new Image(d, "c:\\icons\\open.jpg");
        final Image childIcon = new Image(d, "c:\\icons\\userH.ico");
        final Image cutIcon = new Image(d, "c:\\icons\\cut.jpg");
        final Image copyIcon = new Image(d, "c:\\icons\\copy.jpg");
        final Image pasteIcon = new Image(d, "c:\\icons\\paste.jpg");

        // create and add the button for performing an open operation
        final CoolItem openCoolItem = new CoolItem(coolBar, SWT.NONE);

        final ToolBar fileToolBar = new ToolBar(coolBar,SWT.HORIZONTAL);
        final ToolItem openToolItem = new ToolItem(fileToolBar, SWT.PUSH);
        openToolItem.setImage(openIcon);
        openToolItem.setText("Open");
        openToolItem.setToolTipText("Open");

        final ToolItem saveToolItem = new ToolItem(fileToolBar, SWT.PUSH);
        saveToolItem.setImage(openIcon);
        saveToolItem.setText("Save");
        saveToolItem.setToolTipText("Save");

        fileToolBar.pack();
        Point size = fileToolBar.getSize();
        openCoolItem.setControl(fileToolBar);
    }
}
```

**Example 15-2.** Using a `ToolBar` with a `CoolBar` (continued)

```
openCoolItem.setSize(openCoolItem.computeSize(size.x, size.y));

final CoolItem editbarCoolItem = new CoolItem(coolBar, SWT.PUSH);
final ToolBar editToolBar = new ToolBar(coolBar, SWT.HORIZONTAL);

// create and add the button for performing a cut operation
final ToolItem cutToolItem = new ToolItem(editToolBar, SWT.PUSH);
cutToolItem.setImage(cutIcon);
cutToolItem.setText("Cut");
cutToolItem.setToolTipText("Cut");

// create and add the button for performing a copy operation
final ToolItem copyToolItem = new ToolItem(editToolBar, SWT.PUSH);
copyToolItem.setImage(copyIcon);
copyToolItem.setText("Copy");
copyToolItem.setToolTipText("Copy");

// create and add the button for performing a paste operation
final ToolItem pasteToolItem = new ToolItem(editToolBar, SWT.PUSH);
pasteToolItem.setImage(pasteIcon);
pasteToolItem.setText("Paste");
pasteToolItem.setToolTipText("Paste");
editToolBar.pack();
size = editToolBar.getSize();
editbarCoolItem.setControl(editToolBar);
editbarCoolItem.setSize(editbarCoolItem.computeSize(size.x, size.y));

final CoolItem fontCoolItem = new CoolItem(coolBar, SWT.PUSH);
final Combo fontCombo = new Combo(coolBar, SWT.READ_ONLY | SWT.BORDER);
String[] items = {"Arial", "Courier", "Times New Roman"};
fontCombo.setItems(items);
fontCombo.pack();
size = fontCombo.getSize();
fontCoolItem.setControl(fontCombo);
fontCoolItem.setSize(fontCoolItem.computeSize(size.x, size.y));
fontCoolItem.setMinimumSize(size);

openToolItem.addListener(SWT.Selection, new Listener() {
    public void handleEvent(Event event) {
        System.out.println("Open");
    }
});

saveToolItem.addListener(SWT.Selection, new Listener() {
    public void handleEvent(Event event) {
        System.out.println("Save");
    }
});

cutToolItem.addListener(SWT.Selection, new Listener() {
```

**Example 15-2.** Using a ToolBar with a CoolBar (continued)

```
        public void handleEvent(Event event) {
            System.out.println("Cut");
        }
    });

    copyToolItem.addListener(SWT.Selection, new Listener() {
        public void handleEvent(Event event) {
            System.out.println("Copy");
        }
    });

    pasteToolItem.addListener(SWT.Selection, new Listener() {
        public void handleEvent(Event event) {
            System.out.println("Paste");
        }
    });

    // create the menu
    Menu m = new Menu(s, SWT.BAR);

    // create a File menu and add an Exit item
    final MenuItem file = new MenuItem(m, SWT.CASCADE);
    file.setText("&File");
    final Menu filemenu = new Menu(s, SWT.DROP_DOWN);
    file.setMenu(filemenu);
    final MenuItem openMenuItem = new MenuItem(filemenu, SWT.PUSH);
    openMenuItem.setText("&Open\tCTRL+O");
    openMenuItem.setAccelerator(SWT.CTRL+'O');
    final MenuItem saveMenuItem = new MenuItem(filemenu, SWT.PUSH);
    saveMenuItem.setText("&Save\tCTRL+S");
    saveMenuItem.setAccelerator(SWT.CTRL+'S');
    final MenuItem separator = new MenuItem(filemenu, SWT.SEPARATOR);
    final MenuItem exitMenuItem = new MenuItem(filemenu, SWT.PUSH);
    exitMenuItem.setText("E&xit");

    // create an Edit menu and add Cut, Copy, and Paste items
    final MenuItem edit = new MenuItem(m, SWT.CASCADE);
    edit.setText("&Edit");
    final Menu editmenu = new Menu(s, SWT.DROP_DOWN);
    edit.setMenu(editmenu);
    final MenuItem cutMenuItem = new MenuItem(editmenu, SWT.PUSH);
    cutMenuItem.setText("&Cut");
    final MenuItem copyMenuItem = new MenuItem(editmenu, SWT.PUSH);
    copyMenuItem.setText("Co&py");
    final MenuItem pasteMenuItem = new MenuItem(editmenu, SWT.PUSH);
    pasteMenuItem.setText("&Paste");
```

### Example 15-2. Using a ToolBar with a CoolBar (continued)

```
//create a Window menu and add Child items
final MenuItem window = new MenuItem(m, SWT.CASCADE);
window.setText("&Window");
final Menu windowmenu = new Menu(s, SWT.DROP_DOWN);
window.setMenu(windowmenu);
final MenuItem maxMenuItem = new MenuItem(windowmenu, SWT.PUSH);
maxMenuItem.setText("Ma&ximize");
final MenuItem minMenuItem = new MenuItem(windowmenu, SWT.PUSH);
minMenuItem.setText("Mi&nimize");

// create a Help menu and add an About item
final MenuItem help = new MenuItem(m, SWT.CASCADE);
help.setText("&Help");
final Menu helpmenu = new Menu(s, SWT.DROP_DOWN);
help.setMenu(helpmenu);
final MenuItem aboutMenuItem = new MenuItem(helpmenu, SWT.PUSH);
aboutMenuItem.setText("&About");

// add action listeners for the menu items

openMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Open");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

saveMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Save");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

exitMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.exit(0);
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

cutMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Cut");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});
```

### Example 15-2. Using a ToolBar with a CoolBar (continued)

```
copyMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Copy");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

pasteMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Paste");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

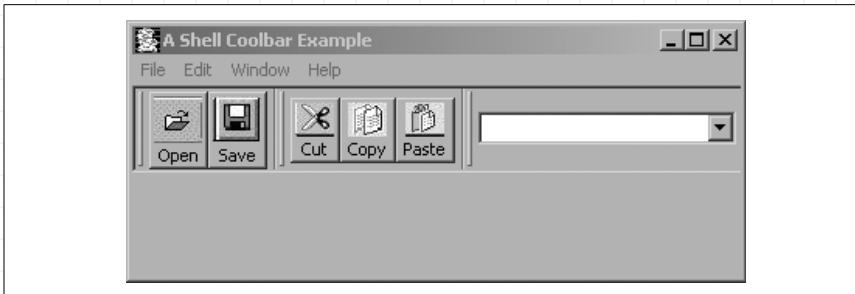
maxMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        Shell parent = (Shell)maxItem.getParent().getParent();
        parent.setMaximized(true);
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

minMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        Shell parent = (Shell)minItem.getParent().getParent();
        parent.setMaximized(false);
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

aboutMenuItem.addSelectionListener(new SelectionListener() {
    public void widgetSelected(SelectionEvent e) {
        System.out.println("Help Invoked");
    }
    public void widgetDefaultSelected(SelectionEvent e) {
    }
});

s.setMenuBar(m);
s.open();
while(!s.isDisposed()){
    if(!d.readAndDispatch())
        d.sleep();
}
d.dispose();
}
```

Executing the new example will show Figure 15-3.



*Looks much  
neater, doesn't  
it? I always use  
toolbars when I  
use a CoolBar*

**Figure 15-3.** CoolBar with ToolBar

## What just happened?

Instead of adding Button objects to the CoolItem objects to create the desired functionality, Example 15-2 creates two ToolBar objects—one for the file functions and the other for the editing functions. The relevant changes are found in this section of code:

```
final CoolItem fileCoolItem = new CoolItem(coolBar, SWT.NONE);

final ToolBar fileToolBar = new ToolBar(coolBar, SWT.HORIZONTAL);
final ToolItem openToolItem = new ToolItem(fileToolBar, SWT.PUSH);
openToolItem.setImage(openIcon);
openToolItem.setText("Open");
openToolItem.setToolTipText("Open");

final ToolItem saveToolItem = new ToolItem(fileToolBar, SWT.PUSH);
saveToolItem.setImage(saveIcon);
saveToolItem.setText("Save");
saveToolItem.setToolTipText("Save");

fileToolBar.pack();
Point size = fileToolBar.getSize();
fileCoolItem.setControl(fileToolBar);
fileCoolItem.setSize(fileCoolItem.computeSize(size.x, size.y));

final CoolItem editbarCoolItem = new CoolItem(coolBar, SWT.PUSH);
final ToolBar editToolBar = new ToolBar(coolBar, SWT.HORIZONTAL);

//create and add the button for performing a cut operation
final ToolItem cutToolItem = new ToolItem(editToolBar, SWT.PUSH);
cutToolItem.setImage(cutIcon);
cutToolItem.setText("Cut");
cutToolItem.setToolTipText("Cut");

// create and add the button for performing a copy operation
final ToolItem copyToolItem = new ToolItem(editToolBar, SWT.PUSH);
copyToolItem.setImage(copyIcon);
```

```

copyToolItem.setText("Copy");
copyToolItem.setToolTipText("Copy");

// create and add the button for performing a paste operation
final ToolItem pasteToolItem = new ToolItem(editToolBar, SWT.PUSH);
pasteToolItem.setImage(pasteIcon);
pasteToolItem.setText("Paste");
pasteToolItem.setToolTipText("Paste");
editToolBar.pack();
size = editToolBar.getSize();
editbarCoolItem.setControl(editToolBar);
editbarCoolItem.setSize(editbarCoolItem.computeSize(size.x, size.y));

```

Here, a CoolItem is first instantiated to serve as the holder for the ToolBar that contains the file items ("Open" and "Save"):

```
final CoolItem fileCoolItem = new CoolItem(coolBar, SWT.NONE);
```

The ToolBar is then created and ToolItem objects are added in the same manner as if the ToolBar were to be used directly on the Shell:

```

final ToolBar fileToolBar = new ToolBar(coolBar, SWT.HORIZONTAL);
final ToolItem openToolItem = new ToolItem(fileToolBar, SWT.PUSH);
openToolItem.setImage(openIcon);
openToolItem.setText("Open");
openToolItem.setToolTipText("Open");

final ToolItem saveToolItem = new ToolItem(fileToolBar, SWT.PUSH);
saveToolItem.setImage(saveIcon);
saveToolItem.setText("Save");
saveToolItem.setToolTipText("Save");

```

Finally, the ToolBar is associated with the CoolItem as if it were any other widget:

```

fileToolBar.pack();
Point size = fileToolBar.getSize();
fileCoolItem.setControl(fileToolBar);
fileCoolItem.setSize(fileCoolItem.computeSize(size.x, size.y));

```

The process is then repeated to create a separate ToolBar for the edit functions ("Cut," "Copy," and "Paste"):

```

final CoolItem editbarCoolItem = new CoolItem(coolBar, SWT.PUSH);
final ToolBar editToolBar = new ToolBar(coolBar, SWT.HORIZONTAL);

//create and add the button for performing a cut operation
final ToolItem cutToolItem = new ToolItem(editToolBar, SWT.PUSH);
cutToolItem.setImage(cutIcon);
cutToolItem.setText("Cut");
cutToolItem.setToolTipText("Cut");

// create and add the button for performing a copy operation
final ToolItem copyToolItem = new ToolItem(editToolBar, SWT.PUSH);
copyToolItem.setImage(copyIcon);
copyToolItem.setText("Copy");

```

```
copyToolItem.setToolTipText("Copy");

// create and add the button for performing a paste operation
final ToolItem pasteToolItem = new ToolItem(editToolBar, SWT.PUSH);
pasteToolItem.setImage(pasteIcon);
pasteToolItem.setText("Paste");
pasteToolItem.setToolTipText("Paste");
editToolBar.pack();
size = editToolBar.getSize();
editbarCoolItem.setControl(editToolBar);
editbarCoolItem.setSize(editbarCoolItem.computeSize(size.x, size.y));
```

As the example demonstrates, you can maintain functional groupings using the `ToolBar` and `CoolBar` in combination. The issue of the user being able to separate functional groupings is eliminated, as is the issue of proper sizing of individual widgets due to the associated images. Finally, the ability to use text labels together with images is restored.

`Coolbars` are a great way to easily permit the user to customize the user interface to her taste; however, care must be taken in planning, in order to maintain proper sizing of the individual widgets placed on the `CoolBar` and in maintaining functional groups as the user rearranges items on the `CoolBar`.

