

A first Java Swing Applet

Most of the information that I see seems to indicate that Java Swing Applets are the current type of Applet that should be used on the Internet. The problem is that the java.sun.com website currently provides little insight in writing Swing Applets. I found one paragraph that states that you must extend the JApplet class to write a java web applet. This note expands on the brief information found at Sun Microsystems using a simple application based on a swing applet embedded in a webpage.

To write a java swing applet you must import both the java.awt.* class library and also the javax.swing.* library. For some reason I also found it necessary to import the javax.swing.JApplet library. If you expect to track any event such as a mouse click or the press of a button you will also need to import the java.awt.event.* library.

```
import java.awt.*;                //import normal applet classes
import java.awt.event.*;          //import event listeners
import javax.swing.*;            //import swing components
import javax.swing.JApplet;      //import swing applet interface

public class Whatever extends JApplet
    implements ActionListener {

    public void init() {           //initialize the applet
        getContentPane().add(label1, BorderLayout.NORTH);
        panel.setLayout(new GridLayout(1,2));
        panel.add(but1);
        panel.add(but2);
        getContentPane().add(panel, BorderLayout.SOUTH);
        but1.addActionListener(this);
        but2.addActionListener(this);
    }
    public void actionPerformed(ActionEvent event) {
        Object source = event.getSource();
        if(source == but1)
            label1.setVisible(true);
        else //if source == but2
            label1.setVisible(false);
    }
    private JLabel label1 = new JLabel("Hello Java Swing World");
    private JPanel panel = new JPanel();
    private JButton but1 = new JButton("ON");
    private JButton but2 = new JButton ("OFF");
}
```

Above I have listed the shell of a simple Java Swing applet called Whatever.java. One thing that I could not find on the Sun Microsystems website is a reference to the init() member function of JApplet. Without this to initialize the JApplet it will not execute properly.

To compile this applet and execute it go to the command prompt and log to the area or disk where you have stored this simple applet. You can create the applet with the NotePad editor found in the accessory section of the Windows Program menu. Make sure that you select "all files" before saving it as Whatever.java

My javac (java compiler) is located in the c:\j2sdk1.4.0_02\bin file folder and I stored the java file on a floppy disk in drive a: so my command line looked like the following when I compiled the program.

```
A:\>c:\j2sdk1.4.0_02\bin\javac Whatever.java
```

This action creates the java class file for this applet called Whatever.class The java class is what executes from the webpage html file.

To execute this applet I created a very simple html file that I called test.html and also stored on my floppy disk in drive a:

```
<APPLET CODE=Whatever.class WIDTH=200 HEIGHT=100>
</APPLET>
```

If you execute the test.html file you will see the applet in action. This simple applet merely states “Hello Java Swing World”. Not really too exciting, but this simple applet is enough to get your trip into the Java Swing world started! The two buttons turn the message on and off, which is used to show how to use the ActionListener class.

Now that you have a simple application (applet) working lets pick it apart to see exactly how it works and what each line of code does in the applet.

The first four lines of the program import all of the java classes that this application uses to accomplish its work.

```
import java.awt.*;                //import normal applet classes
import java.awt.event.*;          //import event listeners
import javax.swing.*;             //import swing components
import javax.swing.JApplet;       //import swing applet interface
```

The java.awt import classes are used to setup a basic java applet while the javax.swing and javax.swing.JApplet import libraries are used to further define the applet as a swing applet. The java.awt.event import is used to gather the ActionListener class so the applet can listen to the button clicks at the bottom of the form.

The last four lines in the program define the various swing components that are used by the program.

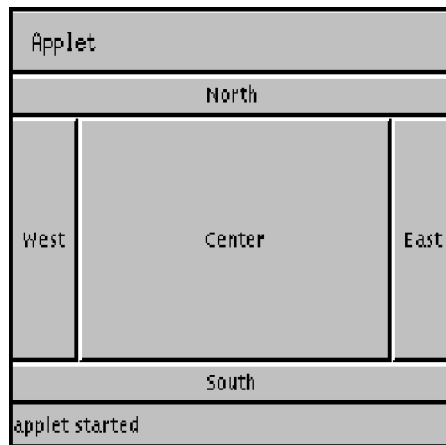
```
private JLabel label1 = new JLabel("Hello Java Swing World");
private JPanel panel = new JPanel();
private JButton but1 = new JButton("ON");
private JButton but2 = new JButton ("OFF");
```

This application uses a swing label called label1 defined as a JLabel with the message “Hello Java Swing World”. Two JButtons are also used for “ON” and “OFF” to turn the label on and off. Finally a JPanel is used to hold the two JButtons. These four items were made private to the Whatever.class so that there scope is only visible from within the applet.

The main member function in the applet is called `init()` and must be present for the applet to function in a web browser.

```
public void init() { //initialize the applet
    getContentPane().add(label1, BorderLayout.NORTH);
    panel.setLayout(new GridLayout(1,2));
    panel.add(but1);
    panel.add(but2);
    getContentPane().add(panel, BorderLayout.SOUTH);
    but1.addActionListener(this);
    but2.addActionListener(this);
}
```

The main frame of a swing applet is by default a `BorderLayout` which has five places for you to attach and view objects (`NORTH` (top of frame), `SOUTH` (bottom of frame), `EAST` (right side of frame), `WEST` (left side of frame), and `CENTER` (center of the frame)). This application attaches things to the `NORTH` and `SOUTH` borders of the frame as a label and a panel. About the only thing you can place on a border without a panel is a label. In this example, we used the `GridLayout` in a panel to place the two pushbuttons on the `SOUTH` border of the frame.



The `GridLayout(3,2)` produces the following alignment in a panel. The first parameter is the number of desired rows and the second parameter is the number of desired columns. Our simple example uses 1 row and 2 columns to place the ON and OFF buttons on the `SOUTH` border of the applet. The `JPanel` is created by using the `add` member function to add the two pushbuttons to the `JPanel` called `pane1`. The entire pane is then added to the `SOUTH` border of the applet again with an `add`. The way we access the main panel is by using the `getContentPane()` member function of the applet.



Finally, we need to implement something called the `ActionListener` which is a class that is used to listen to actions that occur from the mouse for various items in the applet. In our example we need to listen to the ON and OFF buttons so the program can respond to a user as they click on the ON and OFF buttons. To

accomplish this, we add `addActionListener` to each item we which to listen to in the applet as in `but1.addActionListener(this)`. The **this** keyword indicates the current object or instance of the object which is the applet in his case.

```
public void actionPerformed(ActionEvent event) {
    Object source = event.getSource();
    if(source == but1)
        label1.setVisible(true);
    else
        label1.setVisible(false);
}
```

The action listener uses the `actionPerformed` event handler to implement the event (in this case clicking on the buttons). Whenever an event occurs the object that caused an event is returned as a member of `ActionEvent`. In this example we obtain the name of the object in `source` and then use an `if` statement to check and see if `but1(ON)` or `but2(OFF)` caused the event handler to trigger. The outcome determines if `label1` is visible or invisible.