

Creating a New Java Applet Using the Applet wizard

This supplement shows you how to use the Applet wizard to create Java applets with parameters. Suppose you write an applet, as in Example 12.2, "Passing Parameters to Java Applets." The example displays a message on the applet at a specified location. The message and location are passed as parameters using the `<param>` tag in HTML. You can use the Applet wizard to generate the class templates for your applets.

The Applet wizard uses three dialog boxes to collect applet information from the user, and it generates two files: an HTML file and an applet file. You can modify these two files if necessary to make them to work for your project.

The following steps create template files for a new applet:

1. With the project `chapter12.jpr` selected, choose File, New to display the Object Gallery. Click the Applet icon to open the Applet wizard.
2. JBuilder starts Applet wizard (Step 1 of 3) to configure your applet (see Figure 12.7). Edit the Package field to `chapter12` and the Class field to `DisplayMessageApplet`, as shown in Figure 1. Click Next. You will see Step 2 of 3 of the Applet wizard, as shown in Figure 2.

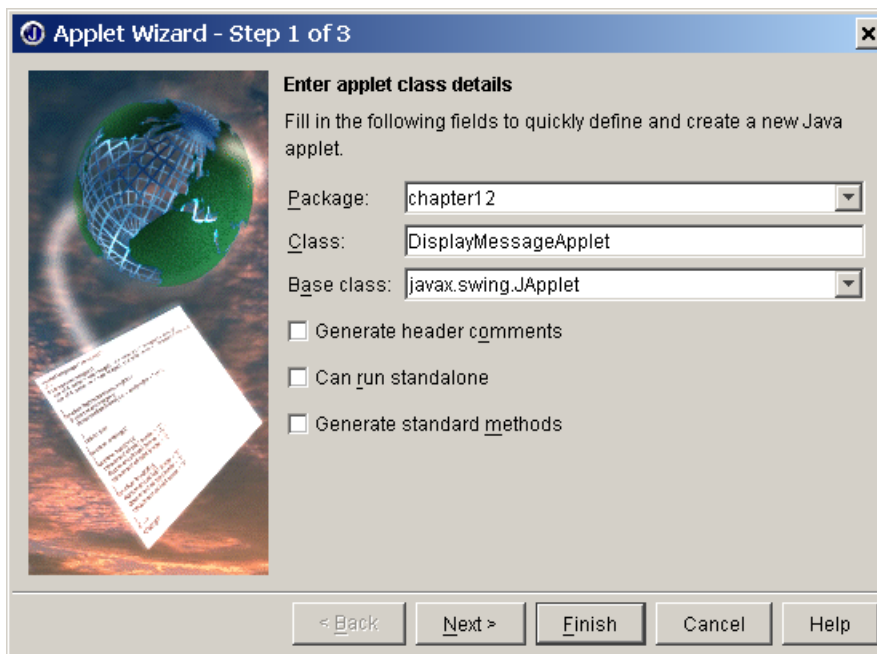


Figure 1

Applet wizard Step 1 of 3 prompts you to enter the package name, the applet class name, and other optional information.

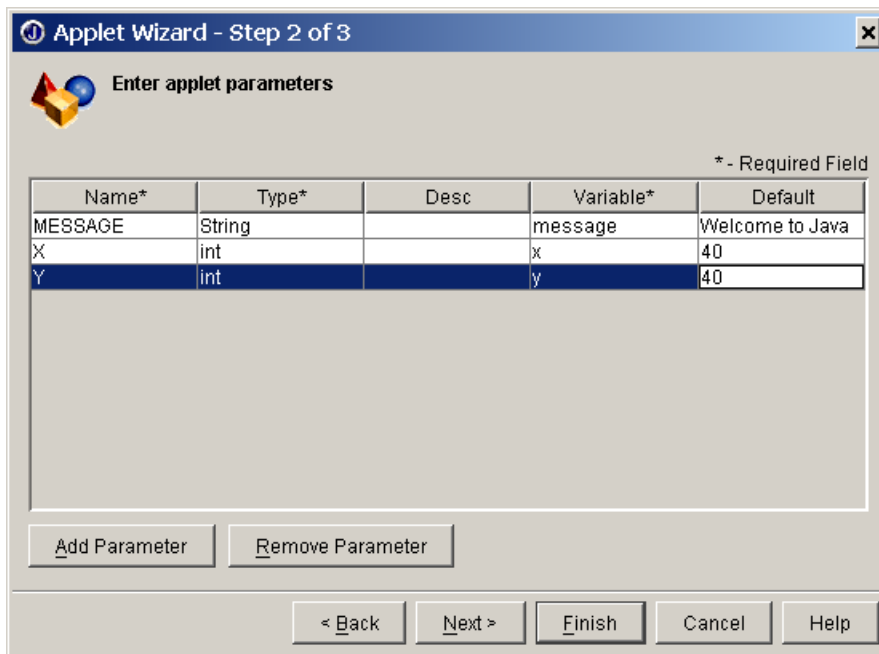


Figure 2

Applet wizard Step 2 of 3 enables you to enter HTML parameters to be passed to the applet.

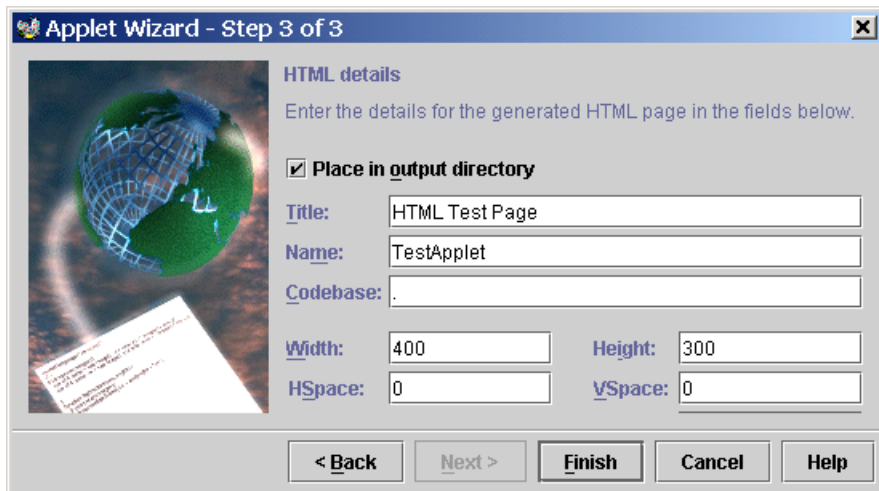


Figure 3

Applet wizard Step 3 of 3 enables you to specify the applet viewing-area size and other properties for the HTML page.

3. Complete Step 2 of 3 in the Applet wizard. This step enables you to define parameters in the HTML file to be passed to your applets. Each parameter is defined in one line. The Name column specifies the name in the `<param>` tag. The Variable column specifies the variable for the parameter in the program. The Type column specifies the variable type in the program.

The Default column specifies the value of the parameter passed from the HTML file. The Description column gives an explanatory description of the parameter.

- 3.1. Add three parameters in the window, as shown in Figure 1. Click the Add Parameter button to start a new parameter line when adding a new parameter.
- 3.2. Click Next. You will see Step 3 of 3 of the Applet wizard, as shown in Figure 3.
4. Complete Step 3 of 3 in the Applet wizard. This step gives you the option to generate an HTML file for the applet.
 - 4.1. Check the Generate HTML Page option. If you check it, the Applet wizard will generate an HTML file. If you don't check it, you have to create an HTML file manually for this applet. If you checked it, you can specify the width and height of the applet viewing area.
 - 4.2. Fill in the other fields, as shown in Figure 3.
 - 4.3. Click Finish.

The Applet wizard generates two files: `DisplayMessageApplet.html` and `DisplayMessageApplet.java`. The source code for `DisplayMessageApplet.html` is shown in the content pane of the AppBrowser in Figure 4. The source code for `DisplayMessageApplet.java` is shown in Figure 5.

NOTE: The HTML file `DisplayMessageApplet.html` is stored in `c:\example` and the Java source file `DisplayMessageApplet.java` is stored in `c:\example\chapter12`. To view the code in `DisplayMessageApplet.html`, you need to choose the Source tab at the bottom of the content pane.

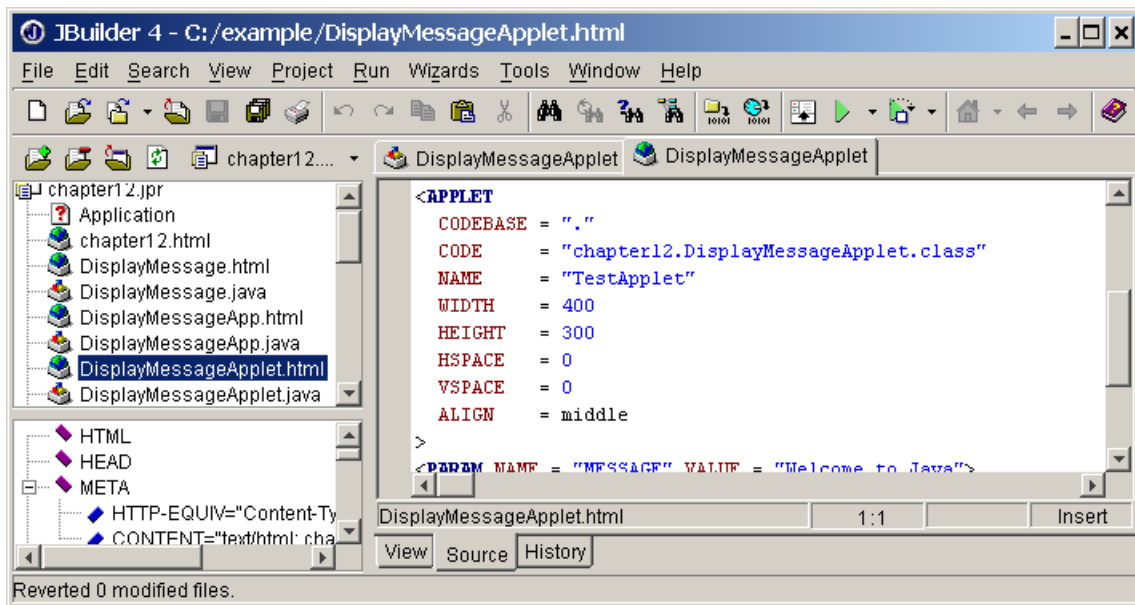


Figure 4

The source code of *DisplayMessageApplet.html* is shown in the *AppBrowser*.

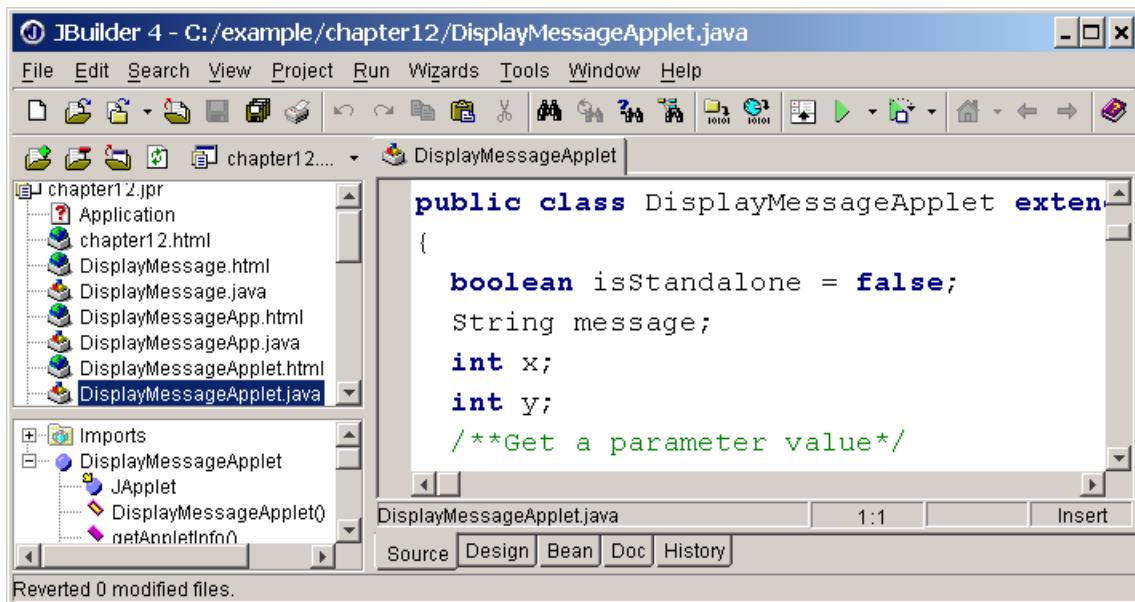


Figure 5

The source code of *DisplayMessageApplet.java* is shown in the *AppBrowser*.

Modifying the Generated Applet Class

Although the Application wizard generates two .java files, the Applet wizard generates just one. In this example, the file is `DisplayMessageApplet.java`. The `DisplayMessageApplet` class contains a constructor and five methods: `getAppletInfo()`, `getParameter()`, `getParameterInfo()`, `init()`, and `jbInit()`. The constructor is useful if you want to run the applet as a standalone application. Because you unchecked the standalone option in Step 1 of 3 of the Applet wizard, however, it generates an empty body for the constructor.

The methods `getAppletInfo()`, `getParameterInfo()`, and `init()` are defined in the `java.applet.Applet` class and are overridden in `DisplayMessageApplet` with concrete implementation. The `jbInit()` method initializes the applet user interface; it is called by the `init()` method. The `getParameter()` method in this class has the same method name as the `getParameter()` method in the `Applet` class. The two methods have different signatures, however. The `getParameter()` method in `DisplayMessageApplet` returns a default value if the parameter does not exist in the HTML file.

NOTE: You can have more methods and varying implementations of the methods if you check certain options in Step 1 of 3 of the Applet wizard.

The variables `message`, `x`, and `y` are generated because you declared them in Step 2 of 3 of the Applet wizard, when specifying the HTML parameters.

To complete the program, add the code for creating an instance of `MessagePanel`, set the appropriate instance properties, and place the instance in the applet. The complete code is shown below (the highlighted lines indicate code manually edited):

```
// DisplayMessageApplet.java: Generated from the Applet wizard
package chapter12;

import java.awt.*;
import java.awt.event.*;
import java.applet.*;
import javax.swing.*;
import chapter10.MessagePanel;

public class DisplayMessageApplet extends JApplet
{
    boolean isStandalone = false;
    String message;
    int x;
    int y;

    /**Get a parameter value*/
```

```

public String getParameter(String key, String def)
{
    return isStandalone ? System.getProperty(key, def) :
        (getParameter(key) != null ? getParameter(key) : def);
}

/**Construct the applet*/
public DisplayMessageApplet()
{
}

/**Initialize the applet*/
public void init()
{
    try
    {
        message = this.getParameter("MESSAGE", "Welcome to Java");
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    try
    {
        x = Integer.parseInt(this.getParameter("X", "40"));
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    try
    {
        y = Integer.parseInt(this.getParameter("Y", "40"));
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
    try
    {
        jbInit();
    }
    catch(Exception e)
    {
        e.printStackTrace();
    }
}

/**Component initialization*/
private void jbInit() throws Exception
{
    this.setSize(new Dimension(400,300));
    MessagePanel messagePanel = new MessagePanel(message);
    messagePanel.setXCoordinate(x);
    messagePanel.setYCoordinate(y);
}

```

```

    getContentPane().add(messagePanel, BorderLayout.CENTER);
    messagePanel.repaint();
}

/**Get Applet information*/
public String getAppletInfo()
{
    return "Applet Information";
}

/**Get parameter info*/
public String[][] getParameterInfo()
{
    String[][] pinfo =
    {
        {"MESSAGE", "String", ""},
        {"X", "int", ""},
        {"Y", "int", ""},
    };
    return pinfo;
}

//static initializer for setting look & feel
static
{
    try
    {
        //UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
        //UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());
    }
    catch(Exception e)
    {
    }
}
}

```

You can view the applet by choosing DisplayMessageApplet.html in the navigation pane. The applet is displayed in the content pane, as shown in Figure 6. If applet is not displayed, choose the View tab in the content pane. You can also run the program by highlighting DisplayMessageApplet.html and then choosing Run, Run Applet in "DisplayMessageApplet.html". An applet is displayed in Figure 7.

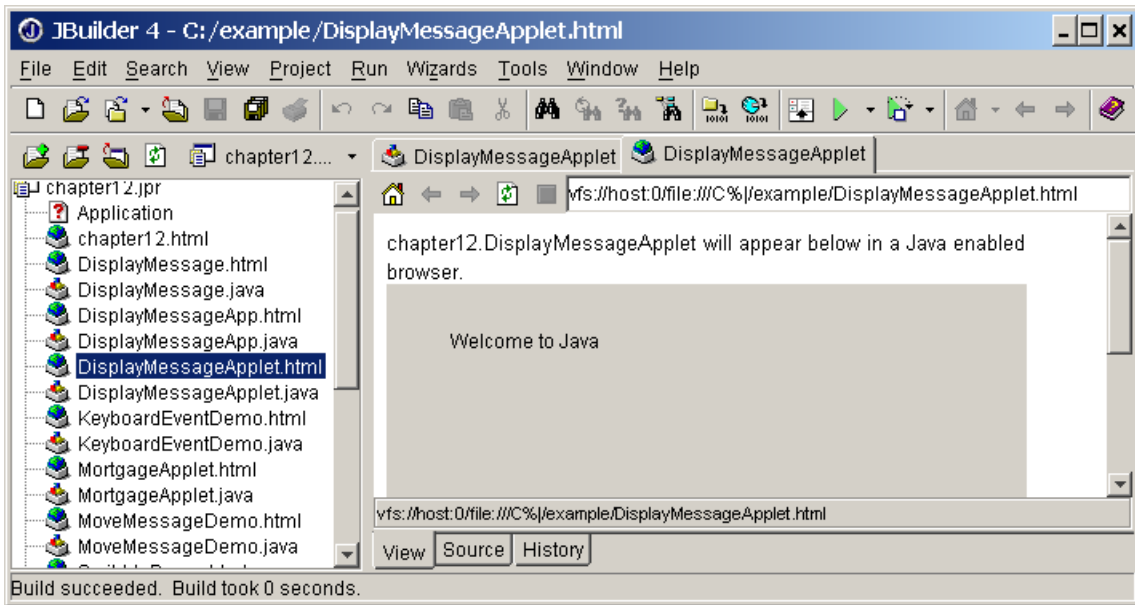


Figure 6

The applet is displayed inside JBuilder.

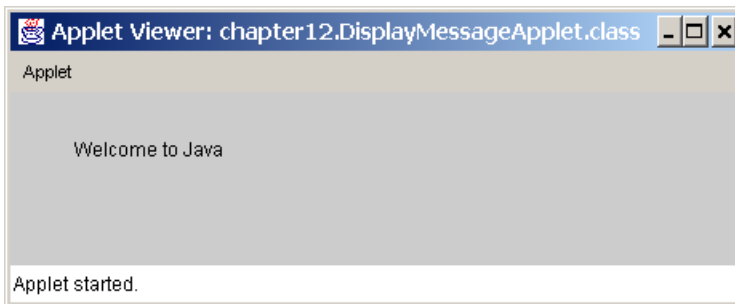


Figure 7

The applet runs outside of JBuilder.