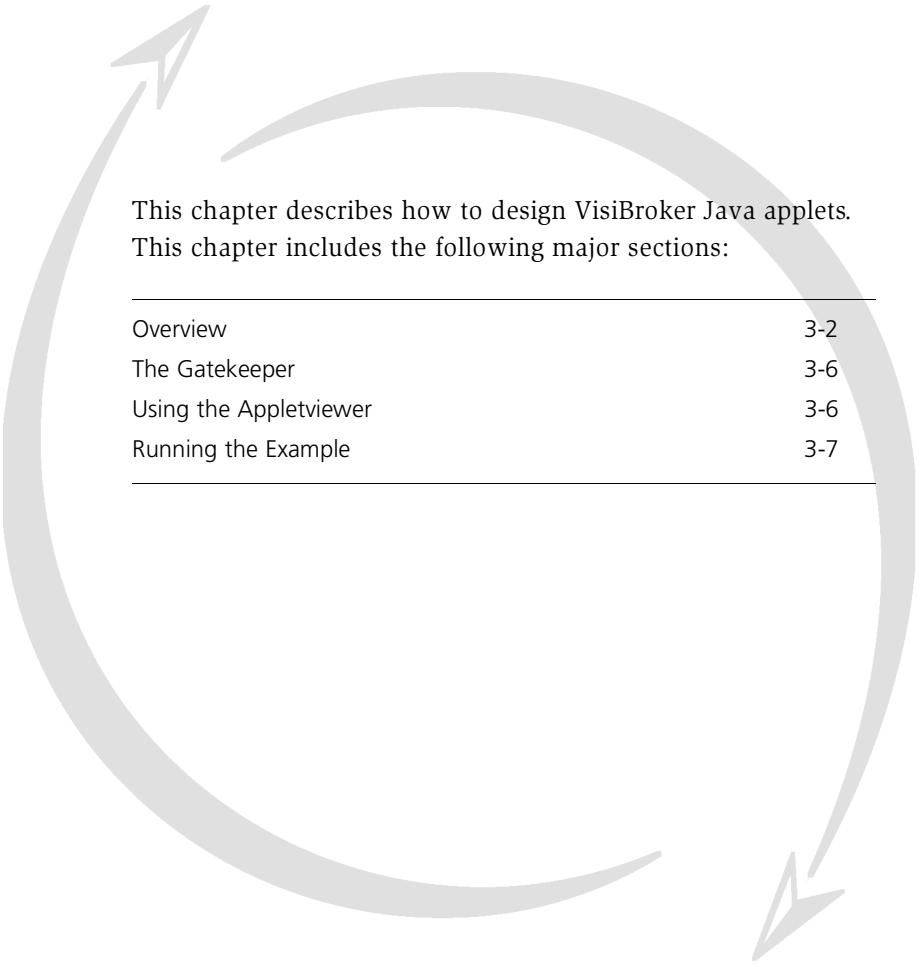
CHAPTER 3

DESIGNING VISIBROKER APPLETS



This chapter describes how to design VisiBroker Java applets. This chapter includes the following major sections:

Overview	3-2
The Gatekeeper	3-6
Using the Appletviewer	3-6
Running the Example	3-7

OVERVIEW

In this chapter, you will extend the banking client application to be an Applet that can be run from any Java-enabled browser using these steps:

- Design a Graphical User Interface (GUI) for the Applet, and
- Write the necessary code to handle GUI events.

The GUI for this example will consist of four widgets that form a two by two grid:

Label(Account Name)	TextField(Account Name)
Button(Check Balance)	TextField(Account Balance)

To cause the *Account Balance* text field to be updated, the user will fill in the *Account Name* text field and then press the *Check Balance* button. The code required to create this GUI is shown in Code Sample 3-1.

CODE

```
setLayout(new GridLayout(2, 2));
add(new Label("Account Name"));
add(_nameField = new TextField());
add(_checkBalance = new Button("Check Balance"));
add(_balanceField = new TextField());
```

Code Sample 3-1 Creating the GUI.

The `_nameField`, `_checkBalance` and `_balanceField` are private variables of the `ClientApplet` class and are used in the handling of the event. The only GUI events that need to be handled in this case are the input of the account name and the pressing of the Check Balance button. The `TextField` widget handles the first case. Most events can be handled by writing an action method on the Applet. For example, code to handle the Check Balance button event will have the following format:

CODE

```
public boolean action(Event ev, Object arg) {
    if(ev.target == _checkBalance) {
      // Request the account manager to open a named account.
      // Get the account name from the name text widget.
      Bank.Account account = _manager.open(_nameField.getText());
      // Set the balance text widget to the account's balance.
      _balanceField.setText(Float.toString(account.balance()));
      return true;
      }
    return false;
}
```

Code Sample 3-2 Handling a button event.

You need to provide the code to an actual call to the bank manager and bank account. The code will get the account name from the first `TextField` widget, use it to open an account, and then fill in the second TextField widget with the account balance.

```
Bank.Account account = _manager.open(_nameField.getText());
_balanceField.setText(Float.toString(account.balance()));
```

The final step is to incorporate this new code into your existing client program. You will also need to write the exception handling code and the ORB initialization code again.

NOTE *The ORB is initialized by passing the* `Applet` *as an argument to the* `ORB_init(Applet)` *method.*

The complete client program is shown Code Sample 3-3. For information about all of the available methods, see the *VisiBroker for Java Reference Manual.*

CODE

```
//ClientApplet.java

import java.awt.*;

public class ClientApplet extends java.applet.Applet {
    private TextField _nameField, _balanceField;
    private Button _checkBalance;
    private Bank.AccountManager _manager;
    public void init() {
      // This GUI uses a 2 by 2 grid of widgets.
      setLayout(new GridLayout(2, 2, 5, 5));
      // Add the four widgets.
      add(new Label("Account Name"));
      add(_nameField = new TextField());
      add(_checkBalance = new Button("Check Balance"));
      add(_balanceField = new TextField());
      // make the balance text field non-editable.
      _balanceField.setEditable(false);
      // Initialize the ORB (using the Applet).
      org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(this);
      // Locate an account manager.
      _manager = Bank.AccountManagerHelper.bind(orb, "BankManager");
    }
    public boolean action(Event ev, Object arg) {
      if(ev.target == _checkBalance) {
        // Request the account manager to open a named account.
        // Get the account name from the name text widget.
        Bank.Account account = _manager.open(_nameField.getText());
        // Set the balance text widget to the account's balance.
        _balanceField.setText(Float.toString(account.balance()));
        return true;
      }
      return false;
    }
}
```

Code Sample 3-3 The complete client applet.

Incorporating the Applet into HTML

The following HTML is used to embed the `Applet` in a simple page:

CODE

```
<h1>VisiBroker Client Applet</h1>
<hr>
<center>
    <applet
      code=ClientApplet.class
      width=200 height=80>
      <h2>You are probably not running a Java-enabled browser,
      Please use a Java-enabled browser (or enable your browser for
      Java)...</h2>
    </applet>
</center>
<hr>
```

Code Sample 3-4 HTML used to embed Applet.

The following screen capture shows how the Applet looks in a browser:

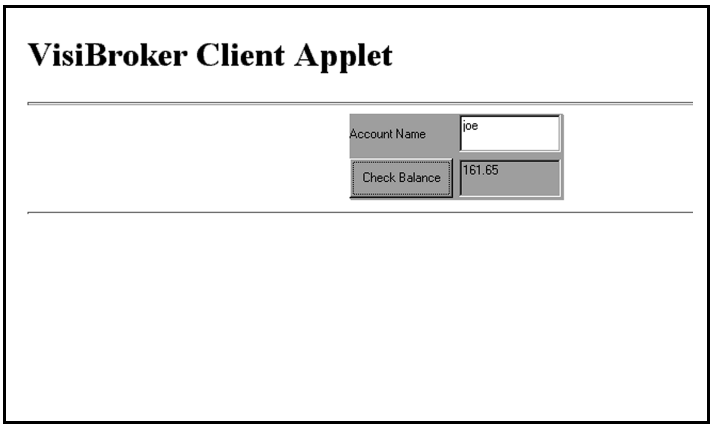


Figure 3-1 The VisiBroker Client Applet as seen in a browser.

This HTML does the following:

- Defines the title of the page.
- Makes a horizontal rule.
- Starts centering mode.
- Starts the `Applet` definition.
- Specifies the code as the class name of the class derived from `Applet` (in this case `ClientApplet.class`).
- Specifies the width and height of the `Applet`, so the HTML page can be laid out correctly.
- Adds a message for non Java-users.
- Completes the `Applet` definition.
- Completes centering mode.
- Makes another horizontal rule.

THE GATEKEEPER

The Gatekeeper enables VisiBroker applets to communicate with object servers across large networks while still conforming to the security restrictions imposed by web browsers. The Gatekeeper may also be used as a web server. For information about the Gatekeeper, refer to *The VisiBroker for Java Gatekeeper Guide*.

USING THE APPLETVIEWER

Now you are ready to run the `ClientApplet`. It is usually best to run the example from the appletviewer before running it from your browser. The appletviewer can be started using the following command:

```
prompt> appletviewer ClientApplet.html
```

If you have trouble running the client applet, you can refer to *The VisiBroker for Java Gatekeeper Guide*

RUNNING THE EXAMPLE

After testing your applet with the appletviewer, you are now ready to run the applet from a Java-enabled browser. Be sure you are using a web browser that supports Java 1.0 release or later. The Uniform Resource Locator (URL) for the applet has the following format:

```
http://<machine-name>[:<port-number>]/<file-name>
```

COMPONENT	DESCRIPTION
<machine-name>	The name of the host that is running the HTTP server.
<port-number>	This should be 15000 if you are using the Gatekeeper as the HTTP server. Otherwise, it should be set to the port number of your HTTP server. If the port number is not specified in the URL, a default port number of 80 is used.
<file-name>	This should be **ClientApplet.html**.

For example, if the **ClientApplet.html** resides in the directory pub/applets on machine foo.bar.com and the Gatekeeper was started from that directory, the URL would be:

```
http://foo.bar.com/pub/applets/ClientApplet.html
```

If you were using the Gatekeeper as your development web server on the machine "foo.bar.com", the URL would be:

```
http://foo.bar.com:15000/ClientApplet.html
```

If the Gatekeeper was started in a directory different than the directory where the server is running, the entire path must be specified and double slashes must be used.

```
http://foo.bar.com:15000//vbroker/java_examples/bank/ClientApplet.html
```

Every time the browser requests a Java class, the IIOP Gatekeeper will print a status message to `stdout`. If the web server encounters errors in loading the file, it will also print error messages to `stderr`.