Lab 6: Creating ActiveX Clients

For background information on this lab, click each of these topics.

Objectives

In this lab, you will use Automation to control and exchange information with Microsoft Excel and Microsoft Access.

After completing this lab, you will be able to use Automation to:

- Open a Microsoft Excel workbook.
- Change information on a worksheet.
- Create a chart from worksheet information.
- Open a Microsoft Access database.
- Use Microsoft Access to open a report based on parameters of your Visual Basic application.

Prerequisites

Before working on this lab, you should be familiar with the following concepts:

- The major constructs of COM
- The major constructs of Automation.
- Fundamentals of the Automation objects in Microsoft Excel.

Lab Setup

To complete this lab, you should have the following setup:

- Visual Basic, version 5.0, or later.

To see a demonstration of the completed lab solution, click this icon.

Estimated time to complete this lab: **60 minutes**

Note There are project and solution files associated with each lab. If you installed the labs during Setup, these files are in the folder <*Install Folder*>\Labs on your hard disk. If you did not install the labs during Setup, you can find them in the \Labs folder of the *Mastering Microsoft Visual Basic 5* CD-ROM.

Exercises

The following exercises provide practice working with the concepts and techniques covered in Chapter 6.

Exercise 1: Controlling Microsoft Excel

In this exercise, you will control Microsoft Excel using Automation.

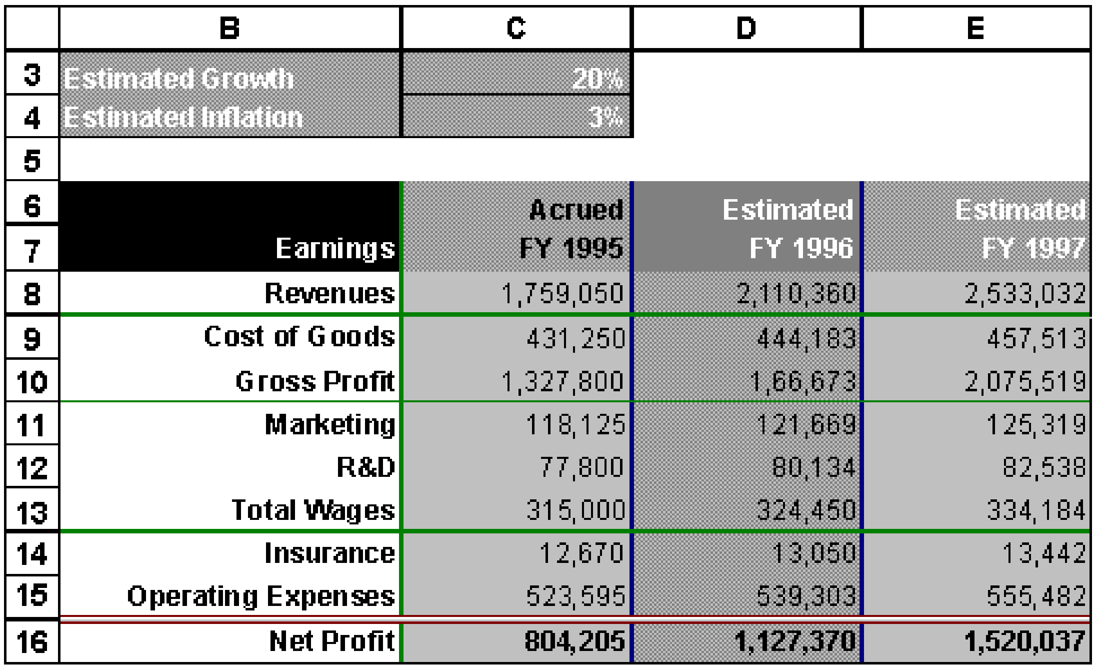
Exercise 2: Creating a Client Application

In this exercise, you will create a client application that uses the functionality of a version of the Credit Card component, which is created in later chapters.

Exercise 1: Controlling Microsoft Excel

In this exercise, you will control Microsoft Excel using Automation. A Microsoft Excel workbook named Earn.xls has been created for you and is located in the *<Install Folder>*\Labs\Lab06 folder.

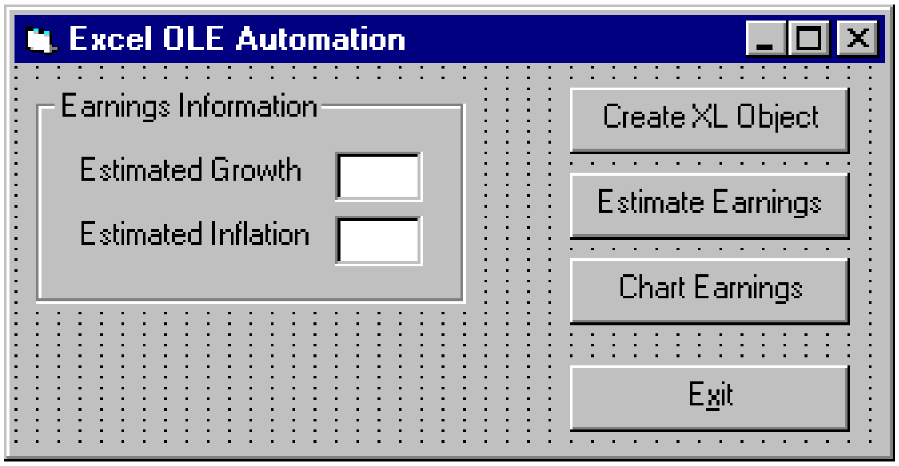
A portion of the workbook is shown in the following illustration. Note that the cells C3 and C4 have been named “Growth” and “Inflation,” respectively. Also, cells C16:E16 have been named “Net_Profit.”



	B	C	D	E
3	Estimated Growth	20%		
4	Estimated Inflation	3%		
5				
6		Acrued	Estimated	Estimated
7	Earnings	FY 1995	FY 1996	FY 1997
8	Revenues	1,759,050	2,110,360	2,533,032
9	Cost of Goods	431,250	444,183	457,513
10	Gross Profit	1,327,800	1,66,673	2,075,519
11	Marketing	118,125	121,669	125,319
12	R&D	77,800	80,134	82,538
13	Total Wages	315,000	324,450	334,184
14	Insurance	12,670	13,050	13,442
15	Operating Expenses	523,595	539,303	555,482
16	Net Profit	804,205	1,127,370	1,520,037

▶ **Create a project**

1. Start a new Visual Basic project.
2. Modify Form1 to resemble the following:



   If you do not want to build the form, you can get a copy of it in the *<Install Folder>* \Labs\Lab06 folder.

3. Save the project as XLAuto.vbp in the \Lab06 folder.

▶ Create an instance of the Microsoft Excel application object

1. Add a reference to the Microsoft Excel 5.0 Object Library.
2. Dimension module-level variables named **xl** and **xlChart**.
3. In the Click event procedure for the **Create XL Object** button, use the **CreateObject** function to create an instance of the Microsoft Excel **Application** object, and assign that object to the variable created in the previous step. For more information see Creating an Instance of Microsoft Excel.
4. Set the **Visible** property of Microsoft Excel to **True**.
5. Use the Microsoft Excel **Open** method to open Earn.xls.

   To see sample answer code, click this icon.

```
Private Sub cmdCreateXLObject_Click()
    set xl = CreateObject("Excel.Application")
    xl.Workbooks.Open App.Path & "\Earn.xls"
    xl.Visible = True
End Sub
```

6. Test the application.
7. Exit.

▶ Send information to Microsoft Excel

1. In the Click event procedure for the **Estimate Earnings** button, activate the Earnings worksheet and set the values of cells Growth and Inflation to the values of the Growth and Inflation text boxes, respectively.

   To see sample answer code, click this icon.

```
Private Sub cmdEstimateEarnings_Click()
    xl.Worksheets("Earnings").Activate
    xl.Worksheets("Earnings").Range("Growth").Value = txtGrowth.Text
    xl.Worksheets("Earnings").Range("Inflation").Value = _
        txtInflation.Text
End Sub
```

2. Test the application.
   - Run the application.
   - Click the **Create XL Object** button to start Microsoft Excel and open the workbook.
   - Switch to Visual Basic, enter percentages in the text boxes, and test the functionality of the **Estimate Earnings** button.
3. Exit Microsoft Excel.

▶ Create a chart

1. In the Click event procedure for the **Chart Earnings** button, if the **xlChart** variable added above is **Nothing**, use the **Select** method to select the named **Range** Net_Profit of the Earnings worksheet.

   Note In these steps, you should use the **xl** module-level variable so it can be used from within all procedures in this exercise.
2. Use the **Add** method of the **Chart** object to create a new chart in the workbook and initialize **xlChart**.
3. Set the **Type** property of the **Chart** object to xl3DColumn. This is a constant in the Microsoft Excel object library.
4. If the **xlChart** variable is not **Nothing**, use the **Activate** method to make it the active sheet.

To see sample answer code, click this icon.

```
Private Sub cmdChartEarnings_Click()
    If xlChart Is Nothing Then
        xl.Worksheets("Earnings").Range("net_profit").Select
        set xlChart = xl.Charts.Add()
        xlChart.Type = xl3DColumn
    Else
        xlChart.Activate
    End If
End Sub
```

5. Test the application.
   - Run the application.
   - Click the **Create XL Object** button to start Microsoft Excel and open the workbook.
   - Switch to Visual Basic and click the **Chart Earnings** button.

▶ **Exit Microsoft Excel**

1. In the Click event procedure for the **Exit** button, use the **Close** method to close the **ActiveWorkbook** object.
2. Use the **Quit** method to close Microsoft Excel.
3. Unload the form and end the application.
4. Test the application.
   - Run the application.
   - Click the **Create XL Object** button to start Microsoft Excel and open the workbook.
   - Switch to Visual Basic and click the **Exit** button.
5. Save the project.

Exercise 2: Creating a Client Application

In this exercise, you will create a client application that uses the functionality of a version of the Credit Card component created in later chapters. This component has been created for you in the folder <*Install Folder*>\Labs\Lab06. This exercise adds an interface to that component.

This version of the Credit Card component exposes two interfaces that provide the following properties and methods:

1. The default interface validates the purchase of an item with these properties and methods.

Name	Type
CardNumber	Property
ExpireDate	Property
PurchaseAmount	Property
Approve	Method

2. The **IManage** interface, which uses the following property, is used to change the credit limit of a client.

Name	Type
CreditLimit	Property

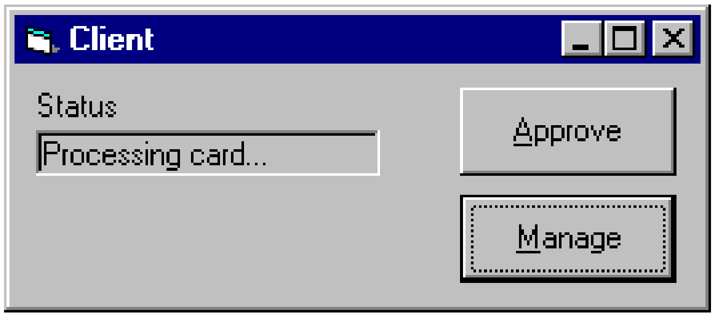
This interface would require some level of security to make use of its functionality.

▶ **Register the component**

1. Locate the file cc.dll, which contains the credit card component.
2. Use RegSvr32.exe to register the component with your system.

▶ **Create a project**

1. Start a new standard project in Visual Basic.
2. Modify the default form to look like the following:



3. Save the form as frmClient in the Lab06 folder.
4. Save the project as CCClient in the Lab06 folder.

▶ **Use the default Automation interface**

1. Add a reference to the CreditCard component.
2. Dimension a form-level variable named **cc** as type Lab.CreditCard, as shown in the following code:

   ```
   Public WithEvents cc as Lab.CreditCard
   ```

   Make sure it can receive events.
3. In the Form_Load handler, create a new instance of the CreditCard component, as shown in this code:

   ```
   Set cc = New Lab.CreditCard
   ```

4. In the Click event procedure for the **Approve** command button, add the following code:

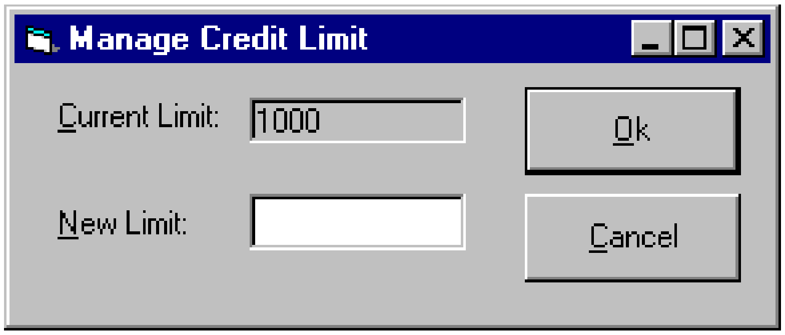
   ```
   cc.ExpireDate = "1/1/99"
   cc.PurchaseAmount = 50
   cc.CardNumber = 1234
   MsgBox cc.Approve
   ```

5. Press F8 to step through the code.

   When you set the **PurchaseAmount** property and call the **Approve** method, you step into the CreditCard project.
6. Try changing the value of **PurchaseAmount** to 2000. Does the **Approve** method return a value of **False**?

▶ **Use the IManage interface**

1. Add a second form, frmManage, that resembles the following:



2. Add a form-level variable of type **IManage**., as shown in this code.

   ```
   Private mng As IManage
   ```

3. In the Form_Load handler, use the CreditCard object variable in frmClient to get a pointer to the **IManage** interface.

   ```
   On Error Resume Next
   Set mng = frmClient.cc
   If mng Is Nothing Then
       MsgBox "The IManage interface is not supported on this object."
       Unload Me
       Exit Sub
   End If
   lblOldLimit = mng.CreditLimit
   txtNewLimit = ""
   ```

4. In the Click event handler of the OK command button, use the **CreditLimit** property of the **IManage** interface to reset the credit limit based on the value provided by the user in the New Limit text box and then unload the form. This code is shown as follows:

   ```
   If IsNumeric(txtNewLimit) Then
       mng.CreditLimit = txtNewLimit
       Unload Me
       Exit Sub
   Else
       txtNewLimit.SetFocus
   End If
   ```

5. In the Click event handler for the **Cancel** command button, unload the form.
6. In the GetFocus event for the **New Limit** text box, add the code to highlight the existing text.

   ```
   Private Sub txtNewLimit_GotFocus()
       txtNewLimit.SelStart = 0
       txtNewLimit.SelLength = Len(txtNewLimit)
   End Sub
   ```

7. Switch to frmClient, and add a handler for the **Manage** command button Click event that shows the frmManage form as a modal window.
8. Save and test the application.
9. Validate the **IManage** interface by setting a breakpoint on frmClient when showing the frmManage form and stepping through the code.

▶ Unregister the CreditCard component

Unregister this version of the CreditCard component to remove the references stored in the registry. This does not modify the DLL itself in any way.

1. Run RegSvr32.exe against the component cc.dll once again, but add the **/u** parameter, as shown in this code:

```
Regsvr32 /u cc.dll
```