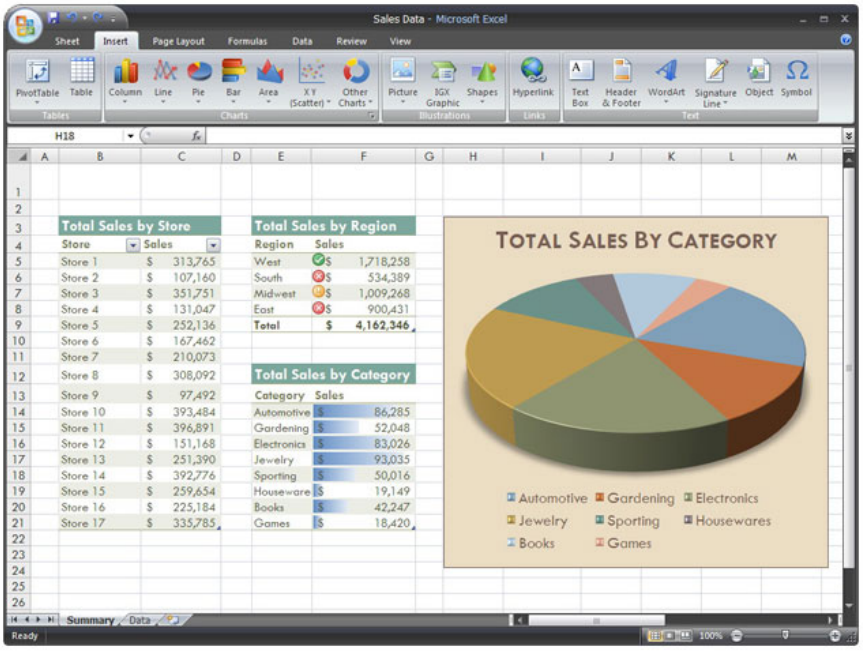
'Packed' Excel file

Many readers require 'security' for Excel files to solve their own problems so that they can 'distribute' professionally. The article will cover the steps to 'pack' and create the interface as a software in VB language and users must install or fill the CD key when opening the file.

Many readers require 'security' for Excel files to solve their own problems so that they can 'distribute' professionally. The article will cover the steps to 'pack' and create the interface as a software in VB language and users must install or fill the CD key when opening the file.



1. In Excel file, you can write some code in VBA language to process data in current Excel file (and other Excel files if you want). These VBA code will retrieve Excel file data through object-oriented model.

For example, you can use **ActiveWorkbook** , **Workbooks** (**name** >) objects to access the current workbook or workbook with the specified name, or **ActiveSheet** or **Sheets** (**name** >) objects to access the current worksheet or worksheet with definite name. For example, the following **CreateSheet** procedure will create a new worksheet (in the current workbook) named " **YourSheet** ", then automatically enter the coordinates of each cell in the cell range from **A1** to **D10** :

```
Sub CreateSheet ()
Dim rg As Range
Dim row As Integer, col As Integer
'Create a new worksheet
```

```
Sheets.Add
' Name the new worksheet
ActiveSheet.Name = " YourSheet "
'set the cell area to be processed
Set rg = Range ("A1: D10")
'Enter the coordinates for each cell
'browse in line
For row = 1 To 10
'browse by column
For col = 1 To 4
rg.Cells (row, col) .Value = "(" & row & "," & col & ")"
Next col
Next row
'save results to file
ActiveWorkbook.SaveAs " c: YourWB.xls "
End Sub
```

In fact, **ActiveWorkbook** , **Workbooks** , **ActiveSheet,** or **Sheets** objects that are accessed in the above procedure are only children of the **Application** object, the Application object describes the running Excel application. Therefore, if you want to use the absolute path to access objects, the procedure **CreateSheet ()** above can be rewritten as follows:

```
Sub CreateSheet ()
Dim rg As Range
Dim row As Integer, col As Integer
'Create a new worksheet
Application.Sheets.Add
'Name the new worksheet
Application.ActiveSheet.Name = " YourSheet "
'set the cell area to be processed
Set rg = Range ("A1: D10")
'Enter the coordinates for each cell
'browse in line
For row = 1 To 10
'browse by column
For col = 1 To 4
rg.Cells (row, col) .Value = "(" & row & "," & col & ")"
Next col
Next row
'save results to file
Application.ActiveWorkbook.SaveAs "c: YourWB.xls"
End Sub
```

Now if you want to convert well-run VBA code into Excel file into a standalone VB 6.0 application, you need to do some of the following:

- Using the " **Microsoft Excel x.0 Object Library** " **library** contains **Excel.Application** objects, **ActiveWorkbook** , **Workbooks** , **ActiveSheet** , **Sheets** . in your VB 6.0 application management project.

- Write code to initialize the Excel.Application object that describes the Excel application, before using this object to retrieve its child objects.

- Copy VBA code using absolute path objects that have run well here.

- After processing data of Excel file, write code to close **Excel.Application** object again.

2. To protect software by illegal users, you can write additional code to check the password, insert this code at the beginning of the software.

To see how to solve the problems we have just presented, we would like to introduce a typical process for building a VB 6.0 application, which will require the user to enter a password, check the password entered with a valid password saved. in the application (or on the data file), if it is not correct, the application will stop, and if the program is correct, it will continue. The program with 1 main form contains a Create button, when the user clicks this button, the application will run Excel, create a new sheet named " **YourSheet** ", enter the coordinates into each cell in the cell range from A1 to D10, save the result to file " **c: YourWB.xls** ":

1. Run VB 6.0, when the New project window is displayed, select the **Standard EXE** icon and the Open button to create a new Project contains a simple Form.

2. Select **Project.Components** menu to display the Components window, browse and select the library named **Microsoft Excel x.0 Object Library** then **OK** to add objects in this library to the ToolBox of the current Project.

3. Draw a button on the form, repeat the attribute (**Name**) = **btnCreate** , attribute **Caption = "Create"** .

4. Double-click the newly created button to create a procedure to handle the event by clicking on the button and then writing the code for this procedure as follows:

```
Option Explicit
'declare the API function to use
Private Declare Sub ExitProcess Lib "kernel32" (ByVal uExitCode As Long)
'defines constants to describe encrypted passwords
Const ENPASS = "zpvsqbtt"
'Simple password password encryption function
Function Encryph (As String) As String
Dim bytes () As Byte
Dim str As String
Dim i As Integer
bytes = StrConv (pass, vbFromUnicode)
str = ""
For i = 0 To UBound (bytes)
str = str & Chr (bytes (i) + 1)
Next i
Encryph = str
End Function

'procedure to initialize the form
Private Sub Form_Load ()
```

```
Dim pass As String
'display password entry form
frmPassword.Show vbModal
pass = frmPassword.txtPassword
Unload frmPassword
'Check the password entered
If ENPASS > Encryph (pass) Then
'If false, stop the application
ExitProcess (1)
End If
'If it is, run normally
End Sub

'service procedure click the Create button
Private Sub btnCreate_Click ()
'define variables to use
Dim Application As Excel.Application
Dim WorkBook As Excel.WorkBook
Dim wksSheet As Excel.Worksheet
Dim rg As Range
Dim row As Integer, col As Integer
'create an Excel application
Set Application = New Excel.Application
'Create the Excel workbook
Set WorkBook = Application.Workbooks.Add

'VBA code is available from Excel file
'Create a new worksheet
Application.Sheets.Add
'Name the new worksheet
Application.ActiveSheet.Name = "YourSheet"
'set the cell area to be processed
Set rg = Range ("A1: D10")
'Enter the coordinates for each cell
'browse in line
For row = 1 To 10
'browse by column
For col = 1 To 4
rg.Cells (row, col) .Value = "(" & row & "," & col & ")"
Next col
Next row
'save results to file
Application.ActiveWorkbook.SaveAs "c: YourWB.xls"

'stop the Excel application
Application.Quit
Set Application = Nothing
End Sub
```

5. Move the mouse to the Project window (usually on the top right of the screen), right-click the **Project** item (the root element of the Project tree) to display the function menu. Select **Add.Form** to display the form creation window, select the **Form** icon and the **Open** button to create a new blank form. Reset properties **Caption = "Enter password using"** , attribute (**Name**) = **frmPassword** .

6. Draw a label, 1 textbox enter the password and an OK button into the form as shown:

Set properties (**Name**) for TextBox as **txtPassword** , attribute (Name) for button is **btnOK** . Double-click the **OK** button to create a procedure for clicking the event on this button and then write the code for the procedure as follows:

```
Private Sub btnOk_Click ()
Me.Visible = False
End Sub
```

7. Select the File.Save Project As . menu to save the Project to disk.

8. Select Run.Start menu to test the application.

9. When the application runs , the form asking to enter the password will be displayed, if the user enters the wrong password specified by the application, the application will stop. If the user enters the correct password, the main form of the application will display, if the user clicks the Create button then the code for creating the Excel object and processing the data will run.