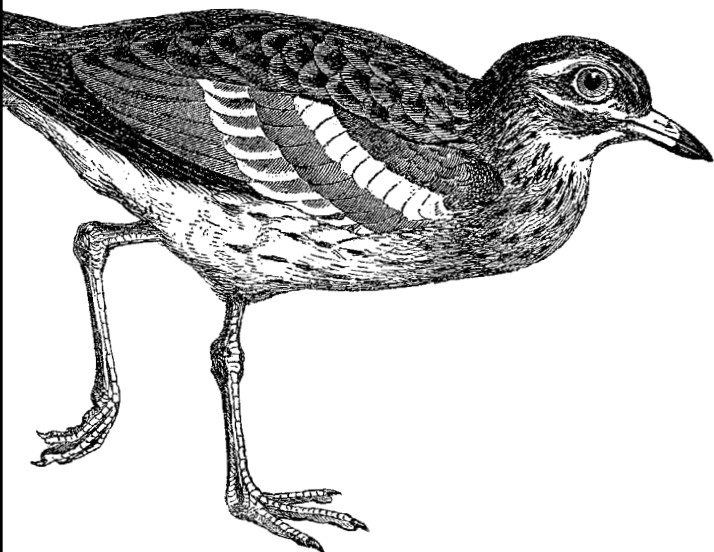


*Up and Running with
Smart Device Development*

.NET Compact Framework

Pocket Guide



O'REILLY®

Wei-Meng Lee

Project A: Currency Converter

This first project shows how to develop a simple currency converter using the .NET Compact Framework's built-in controls. For simplicity, the sample application is limited to three currencies: U.S. Dollar, CNY (Chinese Yuan), and Singapore Dollar. This application downloads the currency exchange rates from a web service when the application is first used. The application stores the rates in an XML document for subsequent usage.

Populating the Form with Controls

In Visual Studio .NET 2003, create a new Smart Device Application (File → New → Project, then select Visual Basic Projects → Smart Device Application). Set the Pocket PC as the target device and select a project type of Windows Application. Your project should open with the default Form1. Before you can add any of the controls, you need to set up two tab forms.

The application in this project contains one TabControl control, which in turn contains two TabPage controls. The first tab page is called Conversion, while the second is called Set Rates. The TabPage control lets you spread the controls across two pages.

To set it up, drag the TabControl from the toolbox onto the form. To add the tab page to a TabControl control, select the TabControl, right-click on it, and select Add Tab (see Figure 3-1).

To change the label of the TabPage, click the tab, select the TabPage (the region just above the tab), right-click on it, and select Properties. Change the Text property of the TabPage to Conversion. Do the same for the second tab page, but set its text to Set Rates. You should enlarge the TabControl so that it fills the form.

You must add the controls listed in Table 3-1 to this form (see Figures 3-2 and 3-3 for the layout). You'll also need to add an Options item to the default MainMenu1.

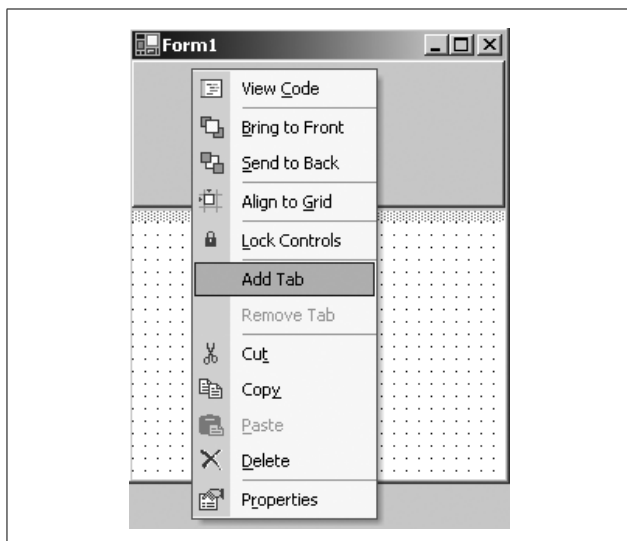


Figure 3-1. Add a new TabPage control

Table 3-1. Controls used in the currency converter

Control type	Conversion tab	Set Rates tab
Label	lblResult	"Base Currency: US\$", "Symbol", "Rate"
TextBox	txtValue	txtRate
ComboBox	ComboBox1, ComboBox2	ComboBox3
Button	cmd0–cmd9, cmdPt, cmdEq, cmdBackSpace, cmdClear	cmdUpdate
ListBox	ListBox1	

The first tab page (Conversion) contains two ComboBox controls for the user to select the currencies to convert. Once the currencies are selected, the user can click on the numbered Button controls to input the amount to convert. The BackSpace button deletes the last character entered, while the Clear button clears the entire line of digits, as displayed

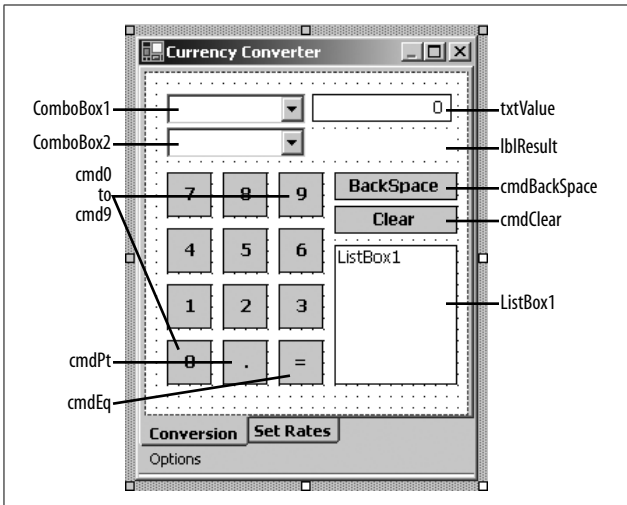


Figure 3-2. Setting up the form (Conversion tab page)

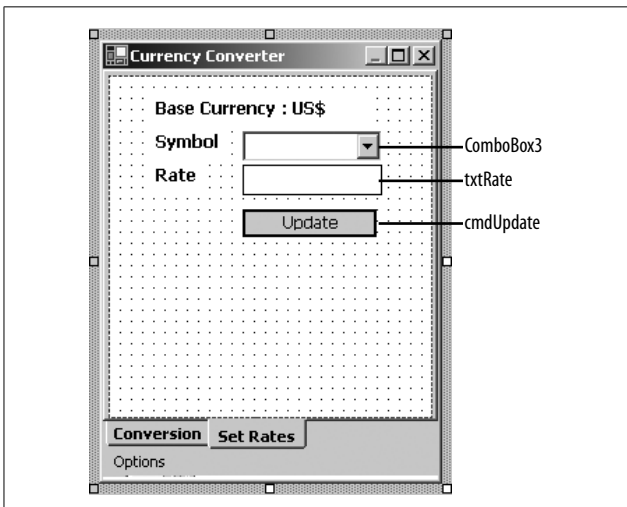


Figure 3-3. Setting up the form (Set Rates tab page)

in the TextBox control. The ListBox control contains a list of previous conversions.

In the second tab page (Set Rates), the user can set the exchange rate for each currency using the ComboBox and TextBox controls.

Populating the Menu

Besides adding the various controls, you must add a MenuItem control to the MainMenu control, as shown in Figure 3-4. When the user selects the Clear History menu item, it clears the ListBox control in the first tab page.

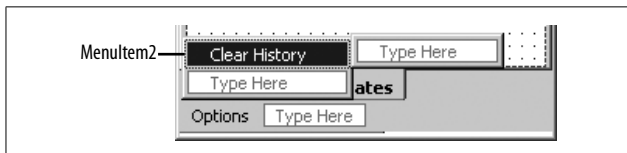


Figure 3-4. Adding the MenuItem control to the MainMenu

Coding the Controls

Now that the controls are set up, you need to begin writing the application logic that performs the currency translation. This application deals with three currencies and uses a single dataset object to bind the exchange rate of each currency to the ComboBox controls, where the end user selects the currencies to convert. To do this, switch to code view and declare the following DataSet object and a global Boolean variable:

```
Dim ds As New DataSet
Dim ready As Boolean = False
```

Now you need to tell the application where to get the exchange rate information. *Rates.xml* contains the exchange rate for each currency:

```
Const FILENAME = "My Documents\Personal\Rates.xml"
```

When the form is first loaded, you need to load the exchange rates by calling the `LoadRates()` method. After that, bind the dataset object using three different `DataView` objects to the three `ComboBox` controls:

```
Private Sub Form1_Load(ByVal sender As Object, _
    ByVal e As System.EventArgs) _
    Handles MyBase.Load
    LoadRates(ds)

    Dim viewFrom As New _
        DataView(ds.Tables("Currency"))
    Dim viewTo As New _
        DataView(ds.Tables("Currency"))
    Dim viewRate As New _
        DataView(ds.Tables("Currency"))

    ComboBox1.Items.Clear()
    ComboBox1.DataSource = viewFrom
    ComboBox1.DisplayMember = "Sym"
    ComboBox1.ValueMember = "Rate"

    ComboBox2.Items.Clear()
    ComboBox2.DataSource = viewTo
    ComboBox2.DisplayMember = "Sym"
    ComboBox2.ValueMember = "Rate"

    ComboBox3.Items.Clear()
    ComboBox3.DataSource = ViewRate
    ComboBox3.DisplayMember = "Sym"
    ComboBox3.ValueMember = "Rate"

    ready = True
End Sub
```

The `LoadRates()` method tries to load the exchange rate information from the file *Rates.xml*, shown here:

```
<?xml version=1.0 standalone=yes?>
<NewDataSet>
  <Currency>
    <Sym>USD</Sym>
    <Rate>1</Rate>
  </Currency>
  <Currency>
    <Sym>CNY</Sym>
```

```

    <Rate>8</Rate>
  </Currency>
</Currency>
  <Sym>SGD</Sym>
  <Rate>1.74</Rate>
</Currency>
</NewDataSet>

```

The base currency is the U.S. dollar. All other currencies are converted relative to the U.S. dollar. For example, \$1 U.S. is equivalent to about 1.69 Singapore dollars. Here is the code for the `LoadRates()` subroutine:

```

Public Sub LoadRates(ByVal ds As DataSet)
  Try
    ds.ReadXml(FILENAME)
  Catch err As Exception
    InitRates(ds)
    ' init the rates if rates.xml
    ' cannot be found
  End Try
End Sub

```

The rates file does not exist when the application is first loaded, so the `InitRates()` method loads the exchange rates from a currency converter web service, such as the one located at <http://www.websvc.com/CurrencyConverter.asmx>. To use this web service—which converts the major currencies in the world—add a Web Reference to your project by right-clicking on your project name in Solution Explorer and then selecting Add Web Reference. The default proxy class name for this web service created by Visual Studio .NET is called `com.websvc.www`.

The `InitRates()` method creates a dataset containing the currency table and three exchange rates obtained from the currency converter web service. If the web service is not reachable, a default exchange rate is used. The information is then written to file:

```

Public Sub InitRates(ByVal ds As DataSet)
  Dim ws As New _

```

```

com.webservicex.www.CurrencyConvertor

ds.Tables.Add("Currency")
ds.Tables("Currency").Columns.Add("Sym")
ds.Tables("Currency").Columns.Add("Rate")

Dim row As DataRow
row = ds.Tables("Currency").NewRow
row("Sym") = "USD"
row("Rate") = 1
ds.Tables("Currency").Rows.Add(row)

row = ds.Tables("Currency").NewRow
row("Sym") = "CNY"
Try
    row("Rate") = ws.ConversionRate( _
        com.webservicex.www.Currency.USD, _
        com.webservicex.www.Currency.CNY)
Catch ex As Exception
    MsgBox("Error contacting web " & _
        "service. Setting to default " & _
        "rate.")
    row("Rate") = 8
Finally
    ds.Tables("Currency").Rows.Add(row)
End Try

row = ds.Tables("Currency").NewRow
row("Sym") = "SGD"
Try
    row("Rate") = ws.ConversionRate( _
        com.webservicex.www.Currency.USD, _
        com.webservicex.www.Currency.SGD)
Catch ex As Exception
    MsgBox("Error contacting web service." & _
        " Setting to default rate.")
    row("Rate") = 1.74
Finally
    ds.Tables("Currency").Rows.Add(row)
End Try
ds.WriteXml(FILENAME)
End Sub

```

Accepting Data Entry

Users enter the amount of currency they wish to convert using the buttons 0 to 9 on the first tab page, which invokes the `cmdButtons()` event. Note that this event is fired whenever one of the 11 buttons (0 to 9 and the decimal point button) is pushed. It must also perform error checking so that users may not enter illegal entries, such as multiple decimal points [.]. Once the button is clicked, the value appends the `TextBox` control (`txtValue`) to show the user which numbers she has entered:

```
Private Sub cmdButtons _  
    (ByVal sender As System.Object, _  
     ByVal e As System.EventArgs) _  
    Handles cmd0.Click, cmd1.Click, cmd2.Click, _  
            cmd3.Click, cmd4.Click, cmd5.Click, _  
            cmd6.Click, cmd7.Click, cmd8.Click, _  
            cmd9.Click, cmdPt.Click  
  
    If txtValue.Text.Length > 7 Or _  
       (txtValue.Text.IndexOf("0") = 0 And _  
        CType(sender, Button).Text = "0") Or _  
       (txtValue.Text.IndexOf(".") >= 0 And _  
        CType(sender, Button).Text = ".") Then  
        ' remove multiple "."  
    Else  
        If txtValue.Text.IndexOf("0") = 0 Then  
            txtValue.Text = ""  
        End If  
        txtValue.Text += CType(_  
            sender, Button).Text  
    End If  
End Sub
```

When the value in the `TextBox` control is changed, indicating that the user has entered another number, the conversion amount must be recalculated. The `UpdateValue()` event handles recalculation when any of these events occur:

- The selection in `ComboBox1` is changed (the currency to convert from).

- The selection in ComboBox2 is changed (the currency to convert to).
- The value in the TextBox control is changed.

Here is the UpdateValue() event:

```
Private Sub UpdateValue _
    (ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles ComboBox1.SelectedIndexChanged, _
    txtValue.TextChanged, _
    ComboBox2.SelectedIndexChanged
    If ready And txtValue.Text <> "" Then
        Try
            lblResult.Text = CSng(txtValue.Text) * _
                CSng(ComboBox2.SelectedValue) / _
                CSng(ComboBox1.SelectedValue)
        Catch ex As Exception
            'do nothing
        End Try
    End If
End Sub
```

The cmdBackSpace_Click() method is invoked when the Back Space button is clicked. It is used to remove the last digit that was entered:

```
Private Sub cmdBackSpace_Click _
    (ByVal sender As System.Object, _
    ByVal e As System.EventArgs) _
    Handles cmdBackSpace.Click

    ' length must be greater than 1 for deleting
    If txtValue.Text.Length > 0 Then
        txtValue.Text = Mid(txtValue.Text, 1, _
            txtValue.Text.Length - 1)
    End If
    ' set to 0 if length is zero
    If txtValue.Text.Length = 0 Then
        txtValue.Text = 0
    End If
End Sub
```

The Clear button resets the TextBox control to the value 0:

```
Private Sub cmdClear_Click _
```

```

        (ByVal sender As System.Object, _
         ByVal e As System.EventArgs) _
        Handles cmdClear.Click
    txtValue.Text = 0
End Sub

```

The “=” button adds the current conversion into the ListBox control:

```

Private Sub cmdEq_Click _
    (ByVal sender As System.Object, _
     ByVal e As System.EventArgs) _
    Handles cmdEq.Click
    Dim str As String
    str = txtValue.Text & ComboBox1.Text & " = " _
        & lblResult.Text & ComboBox2.Text
    ListBox1.Items.Add(str)
End Sub

```

The Clear History menu item simply clears the items in the ListBox control:

```

Private Sub MenuItem2_Click _
    (ByVal sender As System.Object, _
     ByVal e As System.EventArgs) _
    Handles MenuItem2.Click
    ListBox1.Items.Clear()
End Sub

```

The second tab page control for the sample application allows the user to change the exchange rate of each currency. In production, this would frequently be done via the web service, which would grab the latest rates. In this case, users can override the rates included in the *Rates.xml* file. When ComboBox3 control is selected, the rate for the respective currency is displayed:

```

Private Sub ComboBox3_SelectedIndexChanged _
    (ByVal sender As System.Object, _
     ByVal e As System.EventArgs) _
    Handles ComboBox3.SelectedIndexChanged
    txtRate.Text = ds.Tables("Currency").Rows _
        (ComboBox3.SelectedIndex).Item _
        ("Rate").ToString
End Sub

```

The Update button will cause all the dataset objects to be updated and reflect the newly entered rate. It will then call the `saveRate()` method to save the changes to the *Rates.xml* file:

```
Private Sub cmdUpdate_Click _  
    (ByVal sender As System.Object, _  
     ByVal e As System.EventArgs) _  
    Handles cmdUpdate.Click  
    ds.Tables("Currency").Rows _  
        (ComboBox3.SelectedIndex).Item _  
        ("Rate") = txtRate.Text  
  
    saveRate(ds)  
End Sub
```

Finally, the `saveRate()` method saves the new exchange rates into the file:

```
Public Sub saveRate(ByVal ds As DataSet)  
    ds.WriteXml(FILENAME)  
    MsgBox("Rates saved.")  
End Sub
```

That's it! Press F5 in Visual Studio to run the application. Figure 3-5 shows the Currency Converter in action.

A currency converter is just an example of the types of things you can do with .NET Compact Framework. If a currency converter is not what you need, you can extrapolate the logic and tactics shown in this project in thousands of other ways. For example, you've learned:

- How to use various built-in controls in .NET Compact Framework
- How to use datasets to manipulate XML documents
- How to use web services to retrieve information
- How to use XML documents as data storage
- How to program your controls to handle events

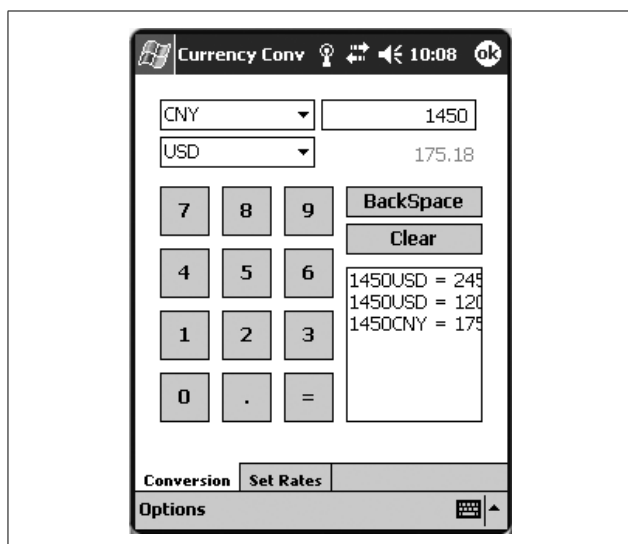


Figure 3-5. Using the Currency Converter