Visual Web Development

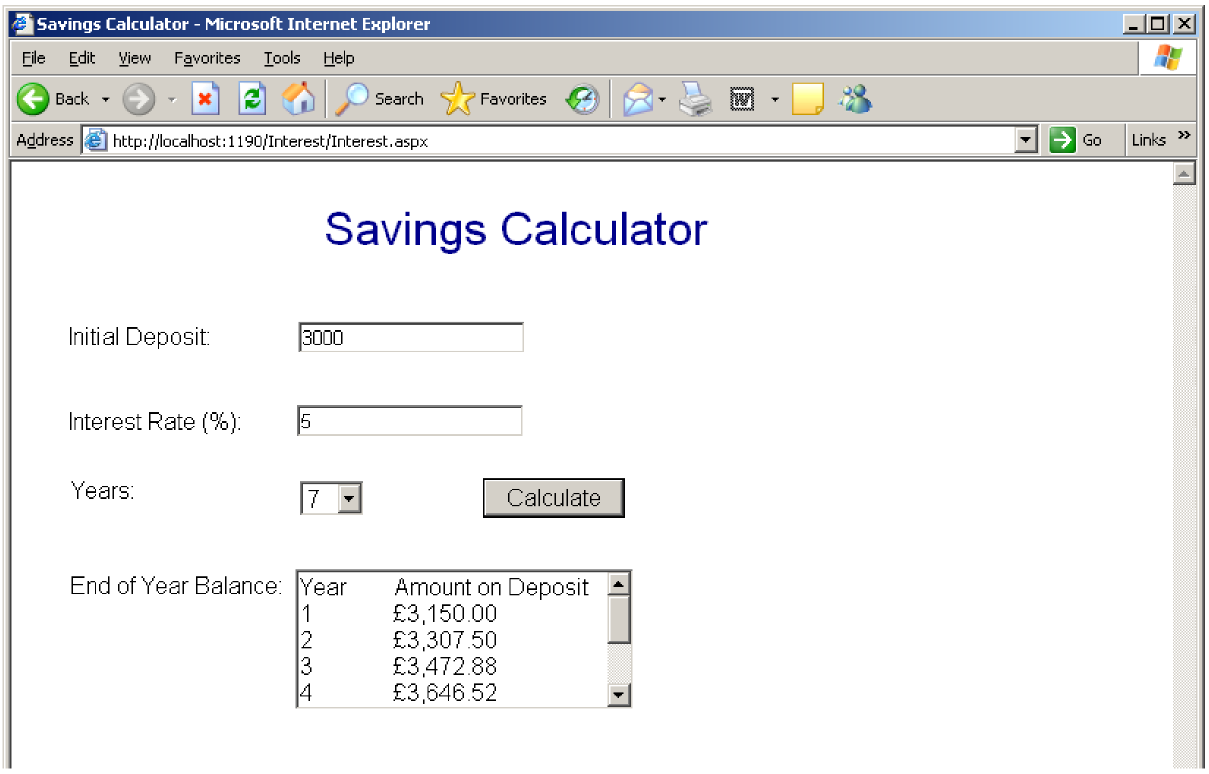
Terry Marris October 2008

9 *Iterations - For ... Next*

In the last chapter we looked at the For Each ... Next loop construct. For Each ... is best used for processing items in a collection of unknown size. Now we look at the For ... Next loop construct, best for situations where we know the exact number of items to be processed.

9.1 Interest Calculator

We show how savings increase over a period of time for a given interest rate.



Savings Calculator - Microsoft Internet Explorer

Address http://localhost:1190/Interest/Interest.aspx

Savings Calculator

Initial Deposit: 3000

Interest Rate (%): 5

Years: 7

Calculate

End of Year Balance:

Year	Amount on Deposit
1	£3,150.00
2	£3,307.50
3	£3,472.88
4	£3,646.52

The user enters 3000, an interest rate of 5% and selects seven years. On pressing Calculate the balance at the end of each year is shown.

9.2 Interest Formula

The formula to calculate how an investment of *deposit* pounds at a fixed *interestRate* grows over several *years* is:

$$\text{amount} = \text{deposit}(1 + \text{interestRate})^{\text{years}}$$

where amount is the final balance in the savings account,

In VB we might write:

```
amount = deposit * (1 + interestRate) ^ years
```

where ^ is the exponentiation operator. So, for example

```
2^3 = 2 * 2 * 2
      = 8
```

(say two to the power of three equals eight).

9.3 For ... Next

We want to produce a table something like:

Year	Amount
1	3150
2	3307
3	3472
4	3646
5	3828

We use a For.. Next loop.

```
For year = 1 to 5
   amount = deposit * (1 + interestRate) ^ year
   ...
Next
```

When year = 1 we calculate the amount at the end of the first year.
When year = 2 we calculate the amount at the end of the second year.
When year = 3 we calculate the amount at the end of the third year.
When year = 4 we calculate the amount at the end of the fourth year.
When year = 5 we calculate the amount at the end of the fifth year.
When year = 6 we finish looping.

9.4 Use Case

Use Case	Interest
Goal	to calculate how an investment grows at a constant interest rate over a number of years.
Pre-condition	numeric values for deposit, interest rate and years are entered.
Post-condition	the value of the investment at the end of each year is shown.
Initiating Actor	the user

Main Success Scenario

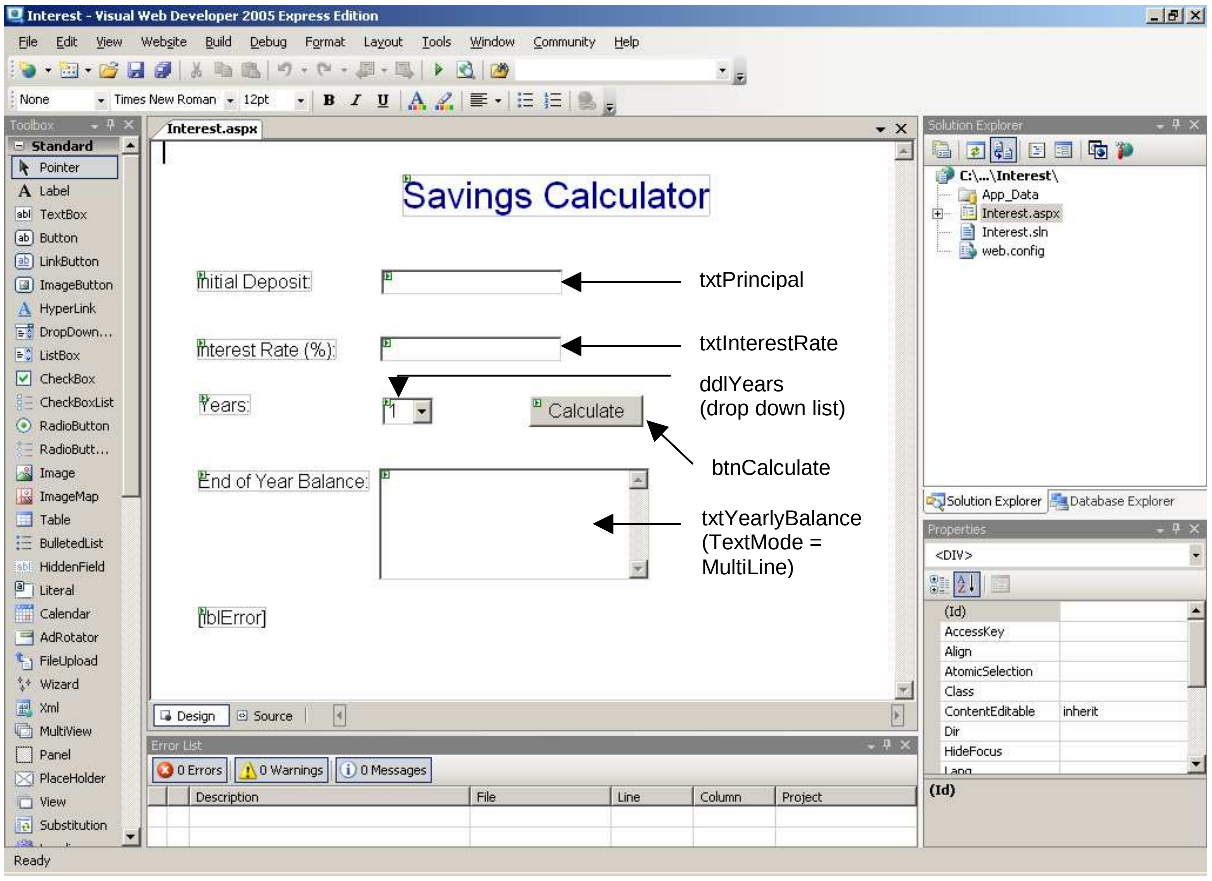
1 user enters initial deposit, interest rate and number of years
2 system calculates the value of the investment at the end of each year
3 exit success

Exceptions

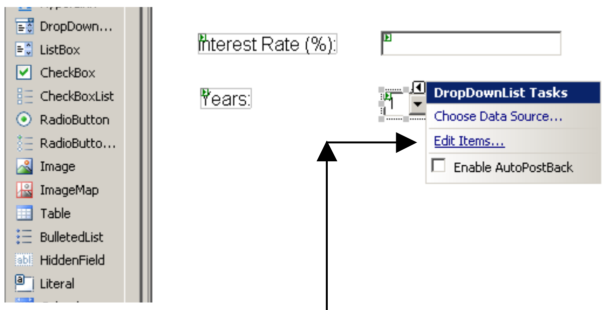
1a non-numeric input
- **1a1** system displays error message
- **1a2** resume 1

1b number too large or too small
- **1b1** system displays error message
- **1b2** resume 1

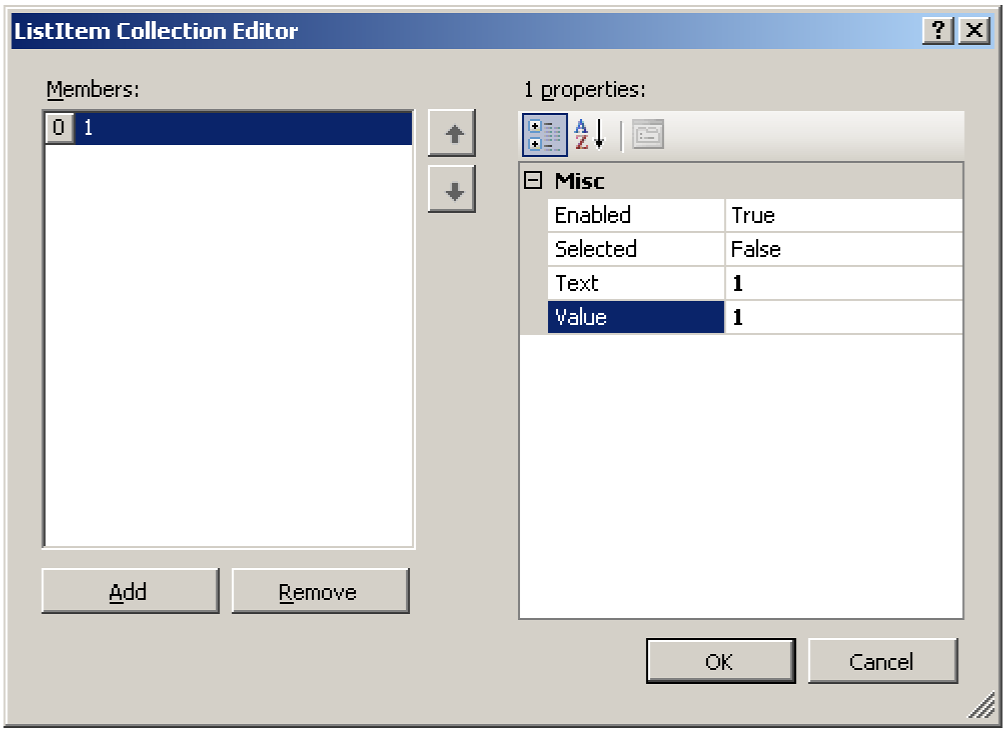
9.5 User Interface



We use a drop down list to present the user with allowable values for year.



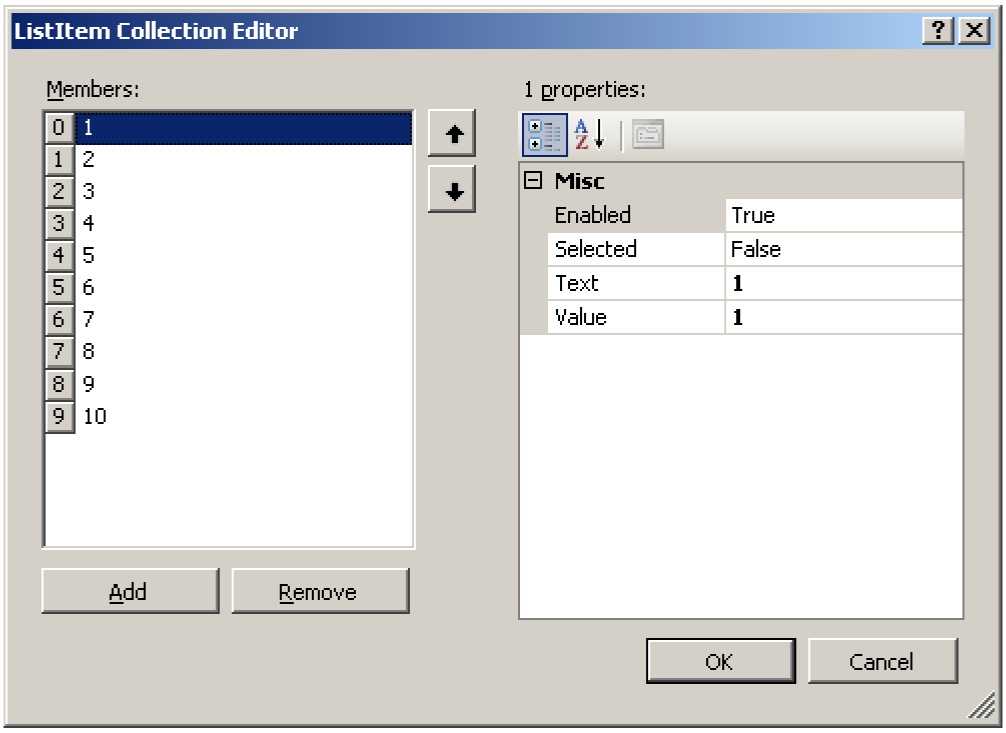
We select Edit Items.



Members	
0	1

Misc	
Enabled	True
Selected	False
Text	1
Value	1

Click on Add, set Text and Value both to 1, OK. Repeat for each integer up to 10.



Members	
0	1
1	2
2	3
3	4
4	5
5	6
6	7
7	8
8	9
9	10

Misc	
Enabled	True
Selected	False
Text	1
Value	1

9.6 The Code

We set the focus on the text box for the principal (i.e. the initial deposit) on start up.

```
Protected Sub Page_Load(ByVal sender As Object, ByVal e AsSystem.EventArgs)
     Handles Me.Load
  txtPrincipal.Focus()
End Sub
```

We declare and initialise our variables. Principal and interest rate are straightforward.

```
Dim decPrincipal As Decimal = Convert.ToDecimal(txtPrincipal.Text)
Dim dblRate As Double = Convert.ToDouble(txtInterestRate.Text)
```

The user selects a value from the drop-down list. This is the selected item and is a ListItem object. We convert the selected item value into an integer.

```
Dim lstItem As ListItem = ddlYears.SelectedItem
Dim intYearSelected As Integer = Convert.ToInt32(lstItem.Value)
```

decAmount is going to change for each year. Initially, it is the amount deposited in the beginning, the principal.

```
Dim decAmount As Decimal = decPrincipal
```

We build the output string line by line. To begin with we have Year, then we add on a Tab, then we add on "Amount on Deposit", then we add on a carriage-return line-feed (like pressing the enter key) ready for the next line of output.

```
Dim strOut As String = "Year" & ControlChars.Tab & "Amount on Deposit" &
ControlChars.CrLf
```

intYearCounter is our loop control variable.

```
Dim intYearCounter As Integer = 0
```

This is where we start looping. The intYearCounter ranges from 1 up to the year selected by the user. Initially, intYearCounter is 1. Each time round the loop the new amount is calculated, formatted to two decimal places and added to the output string, and intYearCounter is increased by 1. Next marks the end of the loop.

```
For intYearCounter = 1 To intYearSelected
  decAmount = decPrincipal * ((1 + dblRate / 100.0) ^ intYearCounter)
   strOut = strOut & (intYearCounter & ControlChars.Tab & _
                    String.Format("{0:c}", decAmount) & ControlChars.CrLf)
Next
```

The entire VB code is shown below.

```
Partial Class _Default
    Inherits System.Web.UI.Page

    Protected Sub Page_Load(ByVal sender As Object, ByVal e As
                System.EventArgs) Handles Me.Load
        txtPrincipal.Focus()
    End Sub

    Protected Sub btnCalculate_Click(ByVal sender As Object, ByVal e As
                System.EventArgs) Handles btnCalculate.Click
        Try
            Dim decPrincipal As Decimal =
                Convert.ToDecimal(txtPrincipal.Text)
            Dim dblRate As Double = Convert.ToDouble(txtInterestRate.Text)
            Dim lstItem As ListItem = ddlYears.SelectedItem
            Dim intYearSelected As Integer = Convert.ToInt32(lstItem.Value)
            Dim decAmount As Decimal = decPrincipal
            Dim strOut As String = "Year" & ControlChars.Tab & "Amount on
                Deposit" & ControlChars.CrLf
            Dim intYearCounter As Integer = 0
            For intYearCounter = 1 To intYearSelected
                decAmount = decPrincipal * ((1 + dblRate / 100.0) ^
                            intYearCounter)
                strOut = strOut & (intYearCounter & ControlChars.Tab & _
                    String.Format("{0:c}", decAmount) & ControlChars.CrLf)
            Next
            txtYearlyBalance.Text = strOut
        Catch ex As Exception
            lblError.Text = ex.ToString()
        End Try
    End Sub
End Class
```

9.7 Exercises

1. Try out the Investment Calculator program shown above.

2. Design, write and test a program that will output a times table chosen by the user.

9.8 Conclusion

In the last chapter we looked at the For Each ... Next loop construct. For Each ... is best used for processing items in a collection of unknown size. In this chapter we looked at the For ... Next loop construct, best for situations where we know the exact number of items to be processed. In the next chapter we look at Do ... While loops.