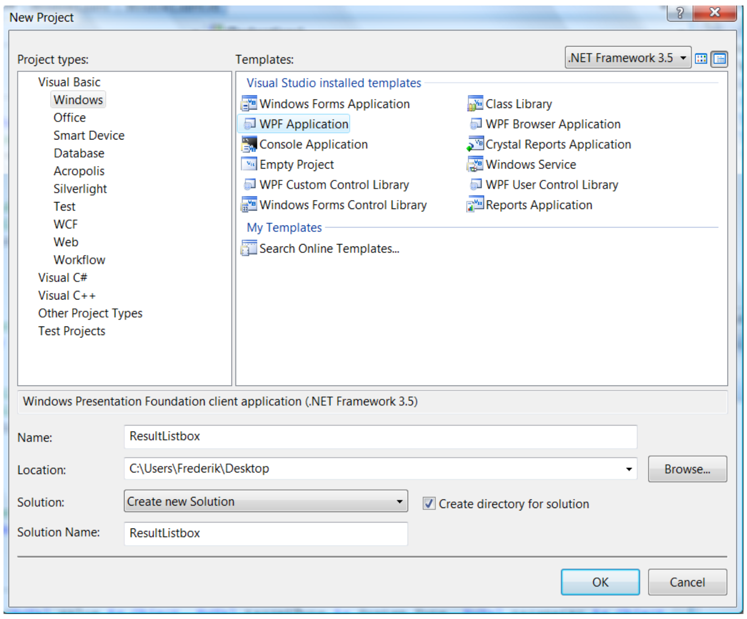
Datatemplate, value converter and ObservableCollection

Step 1: create a new project

I use the beta 2 of Visual Studio 2008 (codename Orcas) and Expression Blend 1.0 for this tutorial.   You can download the beta2 form http://msdn2.microsoft.com/en-us/vstudio/aa700831.aspx. A trial version for Expression Blend can be downloaded from http://www.microsoft.com/expression/products/download.aspx?key=blend.

- Start Visual Studio and choose File → New → Project.
- Choose Visual Basic as language and a WPF Application as template. Also fill in a name for the project and a location.



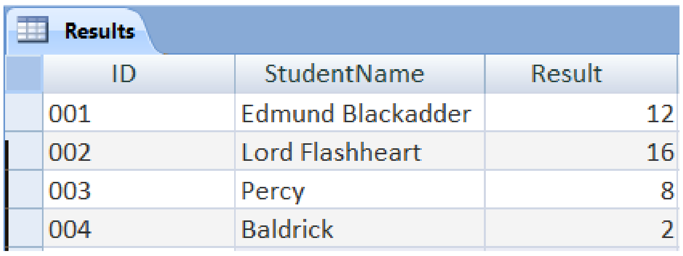
Windows Presentation Foundation client application (.NET Framework 3.5)

Name: ResultListbox
Location: C:\Users\Frederik\Desktop
Solution: Create new Solution
☑ Create directory for solution
Solution Name: ResultListbox

Step 2: create the database

The database is really simple. I got one table called Results who has 3 fields: Id, StudentName and Result. The database can be created with Access 2007, SQL Server, MySql, or any database engine you want.

After creating the database, I added some data to test the project.



Results

ID	StudentName	Result
001	Edmund Blackadder	12
002	Lord Flashheart	16
003	Percy	8
004	Baldrick	2

Step 3: create the class StudentResult

We create a class that describes an object StudentResult. The class contains 3 properties: an id, the name of the student and the result of the student. I have also 2 constructors added.

- Right click the project in the solution explorer and choose Add → Class.
- Name the class StudentResult.vb and add the following code

```
Public Class StudentResult

    'Fields
    Private msID As String
    Private msStudentName As String
    Private miResult As Integer

    'Properties
    Public Property ID() As String
        Get
            Return msID
        End Get
        Set(ByVal value As String)
            msID = value
        End Set
    End Property

    Public Property StudentName() As String
        Get
            Return msStudentName
        End Get
        Set(ByVal value As String)
            msStudentName = value
        End Set
    End Property

    Public Property Result() As Integer
        Get
            Return miResult
        End Get
        Set(ByVal value As Integer)
            miResult = value
        End Set
    End Property

    'Constructor
    Public Sub New()
    End Sub

    Public Sub New(ByVal sId As String, ByVal sStudentName As String,
ByVal iResult As Integer)
        Me.ID = sId
        Me.StudentName = sStudentName
        Me.Result = iResult
    End Sub
End Class
```

Step 4: create the class StudentResultDataAccess

I also need a class to communicate with the database. This class has 2 methods:

- A function GetStudents which returns a list of StudentResult objects
- A method UpdateResult which updates the result of a student in the database. This method receives a StudentResult object as parameter.

This is the code for the class (don't forget to change the connectionstring if you want to run it on your own computer).

```
Imports System.Data
Imports System.Data.OleDb

' This class contains two methods to communicate with the database
' 1. a select statement that gets the data from the database
Public Class StudentResultDataAccess
    ' get all the data needed and return it in a list of StudentResult
objects
    Public Shared Function GetStudents() As List(Of StudentResult)
        ' 1. open the connection with the database
        Dim oCon As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\Frederik\Desktop\ResultListbox\Database.accdb")
        oCon.Open()
        ' 2. create the sql command
        Dim oCommand As OleDbCommand = oCon.CreateCommand
        oCommand.CommandType = System.Data.CommandType.Text
        oCommand.CommandText = "SELECT ID, StudentName, Result FROM
Results"

        ' 3. create the DataAdapter
        Dim oDataAdapter As New OleDbDataAdapter
        oDataAdapter.SelectCommand = oCommand

        ' 4. put the data in a datatable
        Dim oDatatable As New DataTable
        oDataAdapter.Fill(oDatatable)

        ' 5. Loop in the datatable and add every item to a list
        Dim lstStudents As New List(Of StudentResult)
        For Each oDR As DataRow In oDatatable.Rows
            Dim oStudentResult As New StudentResult
            oStudentResult.ID = oDR.Item("ID").ToString
            oStudentResult.StudentName = oDR.Item("StudentName").ToString
            oStudentResult.Result = Convert.ToInt32(oDR.Item("Result"))
            lstStudents.Add(oStudentResult)
        Next

        ' 6. Close the connection with the database and return the list
        oCon.Close()
        Return lstStudents
    End Function

    ' 2. a update statement that updates the result of the student
    Public Shared Sub UpdateResult(ByVal oStudentResult As StudentResult)
        Try
            ' 1. open the connection with the database
            Dim oCon As New
OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\Users\Frederik\Desktop\ ResultListbox\Database.accdb")
            oCon.Open()
            ' 2. create the sql command
            Dim oCommand As OleDbCommand = oCon.CreateCommand
            oCommand.CommandType = System.Data.CommandType.Text
            oCommand.CommandText = "UPDATE Results SET Result=@Result WHERE
ID=@ID"

            ' 3. adding the parameters
            Dim oPar1 As New OleDbParameter("@Result",
oStudentResult.Result)
            Dim oPar2 As New OleDbParameter("@ID", oStudentResult.ID)
            oCommand.Parameters.Add(oPar1)
            oCommand.Parameters.Add(oPar2)

            ' 4. executing the sql statement
            oCommand.ExecuteNonQuery()

            ' 5. close the connection
            oCon.Close()
        Catch ex As Exception
            MessageBox.Show(ex.Message)
        End Try

    End Sub
End Class
```

Step 5: create an ObservableCollection to bind the data

As described in my tutorial Methodbinding, I could use the function GetStudents to bind the data to my listbox. But it is a lot easier to use an ObservableCollection for the data binding with the listbox.

In the same file (StudentResultDataAccess.vb) I create a new class that inherits form ObservableCollection. In the constructor of that class I loop through the list and add every student to the ObservableCollection.

Note: don't forget to import the namespace System.Collections.ObjectModel at the top of the file

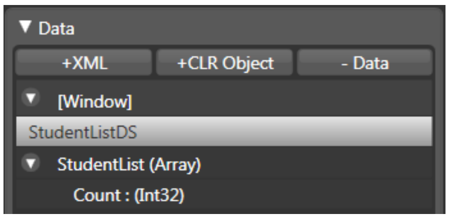
```
' This class returns a observableCollection of StudentResult objects
' The data comes from the database
Public Class StudentList
    Inherits ObservableCollection(Of StudentResult)

    Public Sub New()
        Dim lstStudents As List(Of StudentResult) =
StudentResultDataAccess.GetStudents
        For Each oStudentResult In lstStudents
            MyBase.Add(oStudentResult)
        Next
    End Sub
End Class
```

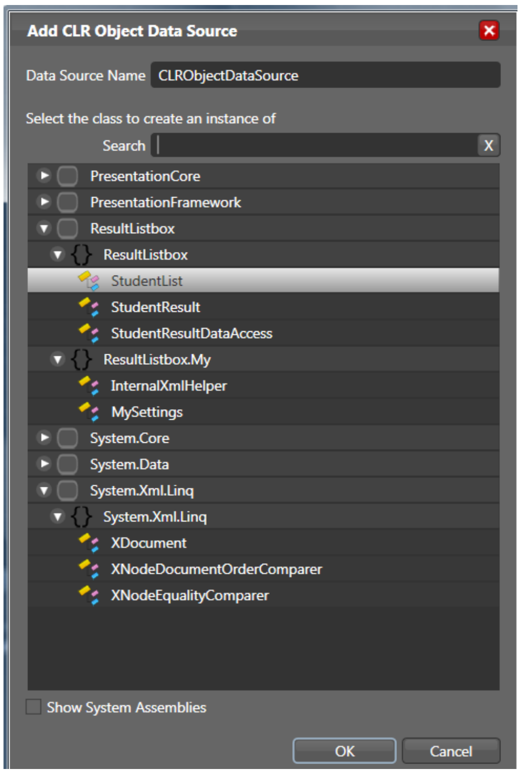
Step 6: bind the ObservableCollection to the listbox

We can bind the ObservableCollection to the listbox with Expression Blend.

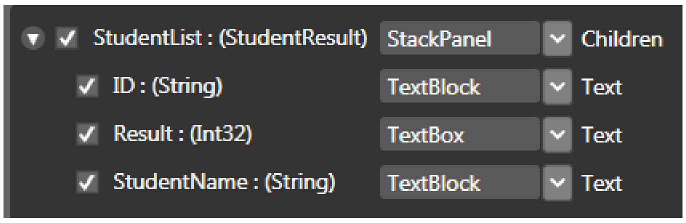
Note: it is a good idea to rebuild the solution before switching between Visual Studio and Expression Blend

- Open the solution in Expression Blend
- Select a listbox from the toolbox and draw it on the stage
- Select the project tab on the right side
- In the data section (at the bottom), click the +CLR Object button





- Choose the StudentList class and click Ok
- You will see that a new dataset is added to the data section
- Right click on the listbox and choose “Bind Itemsource to Data…”
- Select StudentListDS as datasource and StudentList (Array) as fields
- Click “Define DataTemplate” and a new screen will appear
- Choose Textbox instead of Textblock for the field result and click ok



✓ StudentList : (StudentResult)	StackPanel	Children
✓ ID : (String)	TextBlock	Text
✓ Result : (Int32)	TextBox	Text
✓ StudentName : (String)	TextBlock	Text

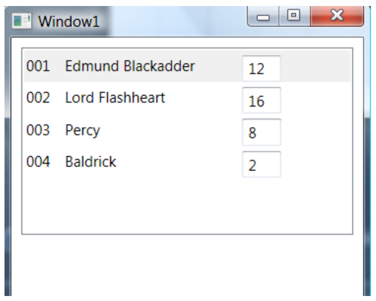
The data is now shown in the listbox. In the next steps we will change the template for the items so that the listbox looks a little better. We will also add a background color to the stackpanel depending on the result.

In the last step, we use two way data binding to sent changes back to the database.

Step 7: Change the datatemplate

- Select the listbox and choose Object → Edit Other Templates → Edit Generated Items (ItemTemplate) → Edit Template from the menu
- Now you can see the differ parts of the listboxitem in the Objects and timeline panel. Make the following changes
    - Select the stackpanel en set the orientation to horizontal. Also set the top margin to 4.
    - In the Objects and timeline panel, select the textbox, right click and choose Order → Bring to Front. Now, the textbox stands on the right side and not in the middle anymore.
    - Select the textblock with the id and set the right margin to 12
    - Select the textblock with the name of the student and set the width to 144
    - Select the textbox for the result and set the width to 32.

You can run the program (press F5) to see the first result. The data comes from the database in the ObservableCollection which is bound to the listbox.



```
001   Edmund Blackadder   12
002   Lord Flashheart     16
003   Percy               8
004   Baldrick            2
```

Step 7: Add a value converter

In this step we will change the background color of the listboxitem to red if the result is lower than 10 or green is the result is higher than 10. In other words, we will bind the text property of the textbox to a brush property of the stackpanel… To make this work, we need a value converter.

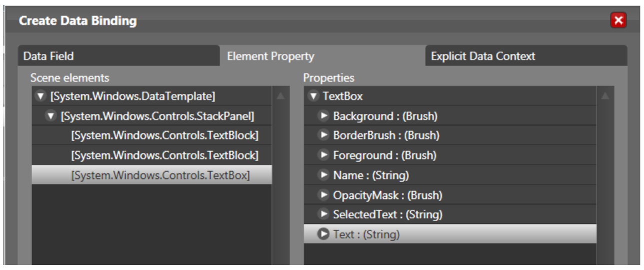
- Add a new class to the project in Visual Studio called ResultToColor.
- Implement the IValueConverter interface. More info about this interface can be found at: http://msdn2.microsoft.com/en-us/library/system.windows.data.ivalueconverter.aspx
- We have two functions: Convert and Convertback. This is the code to convert a text to a color:

```
'This class implements the IValueConverter interface.  The result will
be converted to a brush
' a red brush if the result is lower than 10, or a green brush if the
result is higher

Public Class ResultToColor
    Implements IValueConverter

    Public Function Convert(ByVal value As Object, ByVal targetType As
System.Type, ByVal parameter As Object, ByVal culture As
System.Globalization.CultureInfo) As Object Implements
System.Windows.Data.IValueConverter.Convert
        ' the result is in the argument value
        ' first check that the value is not empty
        If value <> "" Then
            Dim iResult As Integer = value
```

```
      If iResult < 10 Then
        ' return a red SolidColorbrush
        ' first create the red color
        Dim redColor As Color
        redColor.A = 255
        redColor.R = 179
        redColor.G = 6
        redColor.B = 6
        ' second create a brush and return the brush
        Dim redBrush As New SolidColorBrush(redColor)
        Return redBrush
      Else
        ' return a green SolidColorbrush
        ' first create the green color
        Dim greenColor As Color
        greenColor.A = 255
        greenColor.R = 4
        greenColor.G = 132
        greenColor.B = 10
        ' second create a brush and return the brush
        Dim greenBrush As New SolidColorBrush(greenColor)
        Return greenBrush
      End If
    Else
      Return Nothing
    End If
  End Function

  Public Function ConvertBack(ByVal value As Object, ByVal targetType
As System.Type, ByVal parameter As Object, ByVal culture As
System.Globalization.CultureInfo) As Object Implements
System.Windows.Data.IValueConverter.ConvertBack
    ' there is no need to convert back, just return nothing here
    Return Nothing
  End Function
End Class
```

Return back to Expression Blend (don't forget to rebuild the project) and select the stackpanel in the itemtemplate.

- Select the background color property of the stackpanel.
- Click on the little gray rectangle at the right side of the property and choose Data Binding…
- A new window will appear. Select the second tab called "Element property".
- Select the textbox as Scene element (on the left side)
- Select the property text from the property list (on the right side)

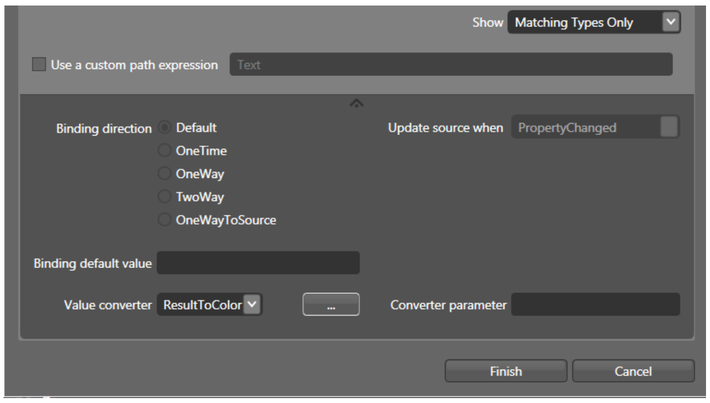

Create Data Binding

Data Field | Element Property | Explicit Data Context

Scene elements
- [System.Windows.DataTemplate]
  - [System.Windows.Controls.StackPanel]
    - [System.Windows.Controls.TextBlock]
    - [System.Windows.Controls.TextBlock]
    - [System.Windows.Controls.TextBox]

Properties
- TextBox
  - Background : (Brush)
  - BorderBrush : (Brush)
  - Foreground : (Brush)
  - Name : (String)
  - OpacityMask : (Brush)
  - SelectedText : (String)
  - Text : (String)

- Click the arrow at the bottom of the window to show the advanced properties.
- Click on the … button near the label Value converter and select the class ResultToColor from the list
- Click Finish at the button of the window.



Show: Matching Types Only

Use a custom path expression: Text

Binding direction: Default / OneTime / OneWay / TwoWay / OneWayToSource

Update source when: PropertyChanged

Binding default value:

Value converter: ResultToColor

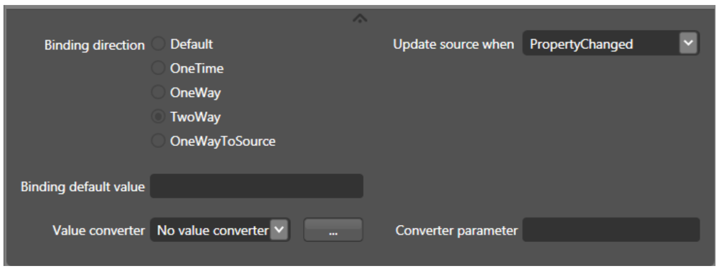
Converter parameter:

Test the program. The students who have a result higher than 10 will have a green background, the others will be red.

Step 7: Two way data binding

If I change the value in the textbox, the value in the database must also change. I use two way data binding to reach this goal.

- Select the textbox in the itemtemplate
- Search for the property text. You will see that there is already an orange border around this property. This means that the property is already bound. We will change the settings of the binding to create two way binding
- Click the little rectangle near the property and choose Data Binding…
- Click the arrow at the bottom to see the advanced settings



Binding direction: Default / OneTime / OneWay / [✓ TwoWay] / OneWayToSource

Update source when: PropertyChanged

Binding default value:

Value converter: No value converter

Converter parameter:

- Set the Binding direction to TwoWay
- Set the Update source when to PropertyChanged
- Click Finish

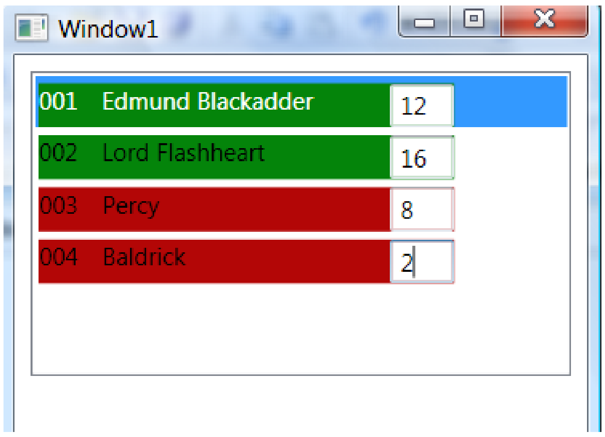
With these settings, changing the value in the textbox will change the property Result in the StudentResult object. This last thing we need to do is call the UpdateResult method when the property changes.

Rebuild the project and go back to Visual Studio. Change the Set part of the Result property to the following:

```
Public Property Result() As Integer
        Get
            Return miResult
        End Get
        Set(ByVal value As Integer)
            miResult = value
            ' update the value in the database
            StudentResultDataAccess.UpdateResult(Me)
        End Set
    End Property
```

Step 8: Test the project

Press F5 to run the project. Change the result of one student, close the program and run it again. The changes should be saved in the database.


Window1

001	Edmund Blackadder	12
002	Lord Flashheart	16
003	Percy	8
004	Baldrick	2