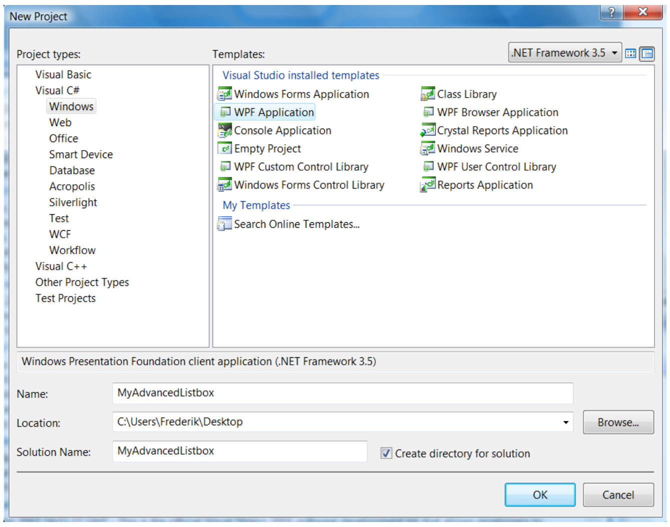
Yes, this is still a listbox!

Step 1: create a new project

I use the beta 2 of Visual Studio 2008 (codename Orcas) and Expression Blend 2.0 September preview for this tutorial.   You can download the beta2 form http://msdn2.microsoft.com/en-us/vstudio/aa700831.aspx. The preview can be downloaded from http://www.microsoft.com/expression/products/features.aspx?key=blend2preview.

- Start Visual Studio and choose File → New → Project.
- Choose Visual C# 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: MyAdvancedListbox

Location: C:\Users\Frederik\Desktop

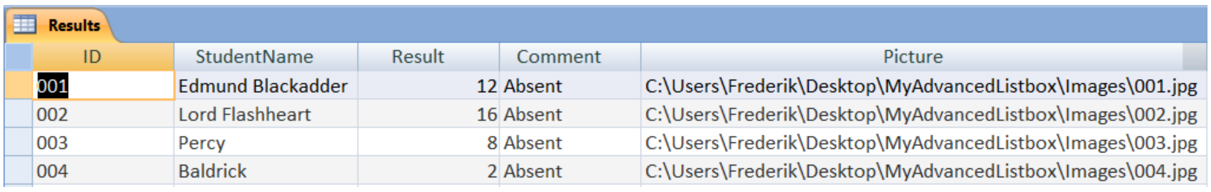
Solution Name: MyAdvancedListbox

☑ Create directory for solution

Step 2: create the database

The database is really simple. I got one table called Results who has 5 fields: Id, StudentName, Result, Comment and Picture . 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	Comment	Picture
001	Edmund Blackadder	12	Absent	C:\Users\Frederik\Desktop\MyAdvancedListbox\Images\001.jpg
002	Lord Flashheart	16	Absent	C:\Users\Frederik\Desktop\MyAdvancedListbox\Images\002.jpg
003	Percy	8	Absent	C:\Users\Frederik\Desktop\MyAdvancedListbox\Images\003.jpg
004	Baldrick	2	Absent	C:\Users\Frederik\Desktop\MyAdvancedListbox\Images\004.jpg

Note: make sure the image exists on the path that you defined in the field Picture.

Step 3: create the class StudentResult

We create a class that describes an object StudentResult. The class contains 5 properties: an id, the name of the student, the result of the student, some comments about the result and the path to an image. I have also 2 constructors added. Finally I override the ToString method with the name of the student

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

```
using System;
using System.Collections.Generic;
using System.Text;

namespace MyAdvancedListbox
{
    public class StudentResult
    {
        // Fields
        private string _id;
        private string _name;
        private int _result;
        private string _comment;
        private string _picture;

        // Properties
        public string Id
        {
            get { return _id; }
            set { _id = value; }
        }

        public string Name
        {
            get { return _name; }
            set { _name = value; }
        }

        public int Result
        {
            get { return _result; }
            set { _result = value; }
        }

        public string Comment
        {
            get { return _comment; }
            set { _comment = value; }
        }

        public string Picture
        {
            get { return _picture; }
            set { _picture = value; }
        }

        // Constructors
        public StudentResult() { }

        public StudentResult(string id, string name, int result, string
comment, string picture)
```

```
            {
                this.Id = Id;
                this.Name = name;
                this.Result = result;
                this.Comment = comment;
                this.Picture = picture;
            }

            // Methods
            public override string ToString()
            {
                return this.Name;
            }
        }
    }
```

Step 4: create the class StudentResultDataAccess and use the Database class

I use a class to communicate with the database. This is the file Database.cs. This class contains several methods to access the database. You can find it in the *.rar file with the entire project.

- Right click the project in the solution explorer and choose Add → Existing item.
- Look for the file Database.cs and add it to the project.
- Right click the project in the solution explorer and choose Properties
- Under the settings section, add a new setting called ConnectionString of the type connectionstring and select your database.

In the StudentResultDataAccess class we create a function which returns a list of StudentResult objects. In the same file I define a new class ResulList that inherits from an ObservableCollection. In the constructor of this class I call the function getStudentsResults and add every item form this list to the ObservableCollection.

The code looks like this:

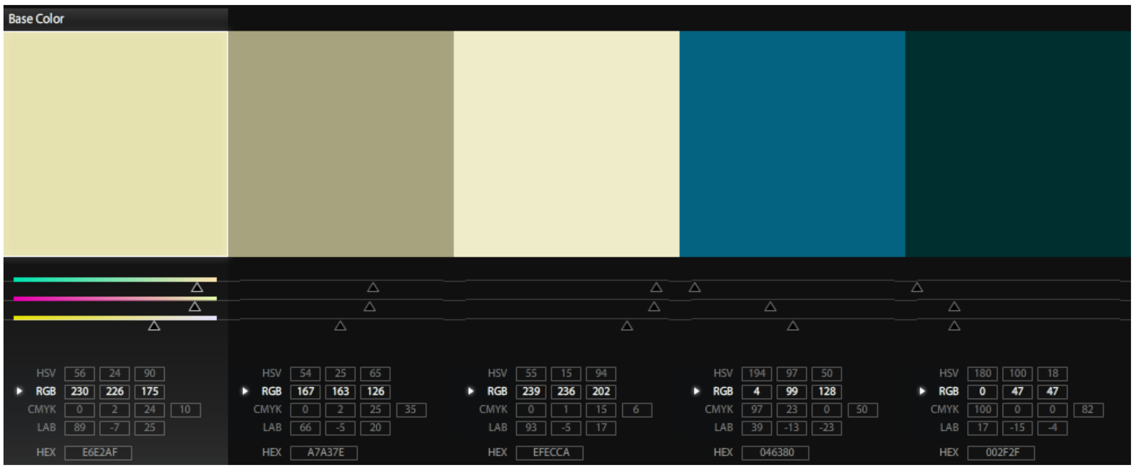
```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Data;
using System.Data.OleDb;
using System.Text;

namespace MyAdvancedListbox
{
    class StudentResultData
    {
        public static List<StudentResult> getStudentResults()
        {
            try
            {
                List<StudentResult> resultList = new List<StudentResult>();
                string sql = "SELECT Id, StudentName, Result, Comment,
Picture FROM Results ORDER BY Id";
                DataTable dt = Database.GetDT(sql);

                foreach (DataRow dr in dt.Rows)
                {
                    StudentResult sr = new StudentResult();
                    sr.Id = Convert.ToString(dr["ID"]);
                    sr.Name = Convert.ToString(dr["StudentName"]);
                    sr.Result = Convert.ToInt32(dr["Result"]);
                    sr.Comment = Convert.ToString(dr["Comment"]);
                    sr.Picture = Convert.ToString(dr["Picture"]);

                    resultList.Add(sr);
                }
                return resultList;
            }
            catch (Exception ex)
            {
                throw ex;
            }
        }
    }

    public class ResultList: ObservableCollection<StudentResult>
    {
        public ResultList()
        {
            List<StudentResult> resultList =
StudentResultData.getStudentResults();
            foreach(StudentResult sr in resultList)
            {
                base.Add(sr);
            }
        }
    }
}
```

Step 5: Add brush resources to the project

Open your project with Expression Blend and do the following:

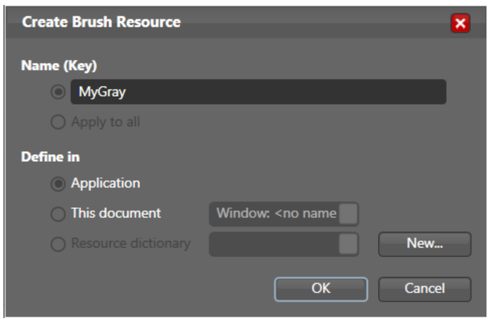
- Set the size of the window element to 800 by 600
- Draw a listbox on the stage. The listbox fills the entire stage with a margin of 8px on each side.
- Set the background of the listbox to no brush and the border to the brush MyYellow. Set the border thickness to 8 on each side.
- Name the listbox lstResults

Now we will add some resource brushes to the project, so we have a nice color scheme. The website http://kuler.adobe.com/ offers a nice tool to pick the “correct” colors.


Base Color

HSV	56 24 90	54 25 65	55 15 94	194 97 50	180 100 18
RGB	230 226 175	167 163 126	239 236 202	4 99 128	0 47 47
CMYK	0 2 24 10	0 2 25 35	0 1 15 6	97 23 0 50	100 0 0 82
LAB	89 -7 25	66 -5 20	93 -5 17	39 -13 -23	17 -15 -4
HEX	E6E2AF	A7A37E	EFECCA	046380	002F2F

- Open your solution with expression blend
- Select the element Window in the objects and timeline panel
- Select the property background in the properties panel
- Insert the RGB values you want
- Press the button Brush to convert the brush to a resource
- Choose a name for the brush and define it in the application so you can reuse it in different windows.


Create Brush Resource

Name (Key)

MyGray

Apply to all

Define in

[✓ Application]

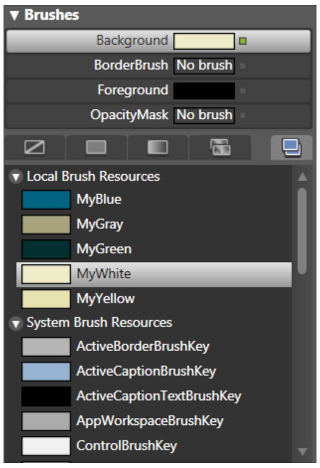
This document — Window: <no name

Resource dictionary

Do this for all the colors you want. You can also do this manually in the app.xaml file:

```
  <Application.Resources>
                    <SolidColorBrush x:Key="MyYellow" Color="#FFE6E2AF"/>
                    <SolidColorBrush x:Key="MyGray" Color="#FFA7A37E"/>
                    <SolidColorBrush x:Key="MyWhite" Color="#FFEFECCA"/>
                    <SolidColorBrush x:Key="MyBlue" Color="#FF046380"/>
                    <SolidColorBrush x:Key="MyGreen" Color="#FF042F2F"/>
  </Application.Resources>
```

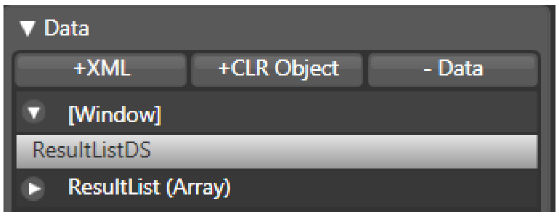
After you added al the resource brushes, set the background for the window to MyWhite.



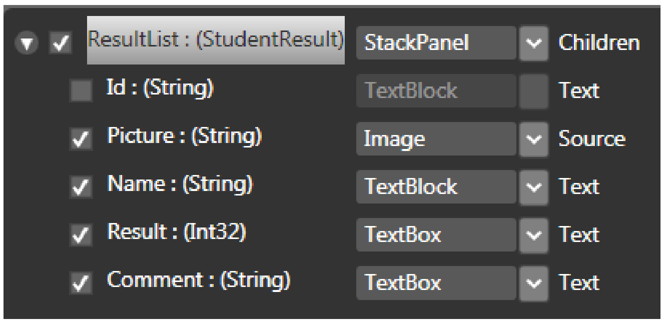
▼ Brushes
Background
BorderBrush No brush
Foreground
OpacityMask No brush
Local Brush Resources
MyBlue
MyGray
MyGreen
MyWhite
MyYellow
System Brush Resources
ActiveBorderBrushKey
ActiveCaptionBrushKey
ActiveCaptionTextBrushKey
AppWorkspaceBrushKey
ControlBrushKey

Step 6: bind the data to the listbox

- Go to the project tab.
- Under the data section, click the +CLR Object button.
- Select the item ResultList



- Right click the listbox on the stage and choose Bind Itemssource to data.
- Select the ResultList (Array) and click on the button Define Datatemplate
- Set the settings just like in the screenshot and click Ok.



✓	ResultList : (StudentResult)	StackPanel	Children
	Id : (String)	TextBlock	Text
✓	Picture : (String)	Image	Source
✓	Name : (String)	TextBlock	Text
✓	Result : (Int32)	TextBox	Text
✓	Comment : (String)	TextBox	Text

The data is now bound to the listbox, but the items look pretty ugly. We will change this in the next step.

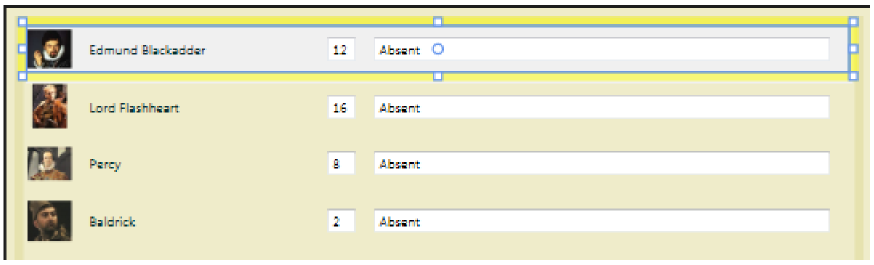
Step 7: Define our own ControlTemplate

In this step, we will create our own template for the listboxitems, so that they look just the way we want.

- Select the listbox in the objects and timeline panel
- Go to Object → Edit other Templates → Edit Generated Items (ItemTemplate) → Edit Template

Now, you can edit the template of the listboxitems.

- Select the Stackpanel and set the orientation to Horizontal and set the background to nothing
- Set the VerticalAlignment for all the items under the stackpanel to center
- Set the foreground of the textblock to MyGreen
- Make the stackpanel bigger and put the controls where you want them. Just like in the screenshot for example.

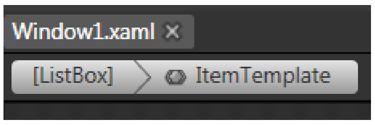


Edmund Blackadder	12	Absent
Lord Flashheart	16	Absent
Percy	8	Absent
Baldrick	2	Absent

Note: to align all the controls on the same line, give the controls a width in pixels instead of Auto. Make sure that the width of the stackpanel is set to auto.

If we run the program, we already have a nice looking listbox. But if you select an item, you see an ugly blue border. We will change this to our blue border.

- Leave the template by clicking on [Listbox] at the top of the screen.



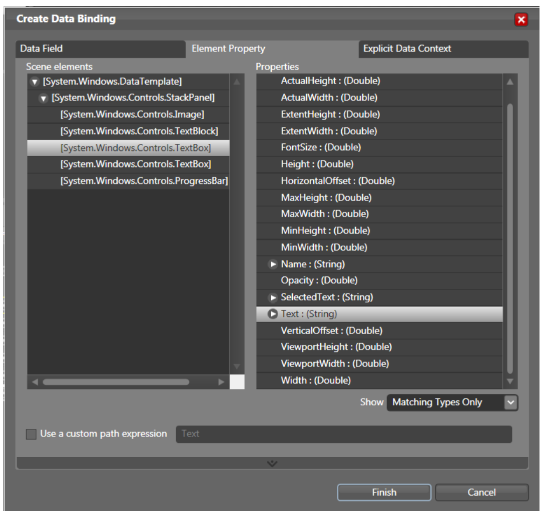
- Select the element listbox in the objects and timeline panel
- Go to Object → Edit other styles → Edit ItemContainerStyle → Edit a copy
- Choose a name for the style and define it in the application. So you can reuse it in a other listbox
- Right click the element style in the objects and timeline panel and choose Edit Control Parts (Template) → Edit template
- Reset the databinding on the backcolor property of the border by clicking the yellow square near the property and choose reset.
- Click on the isSelected trigger
- Change the background property of the border to MyBlue
- Do the same for the trigger IsSelected AND IsSelectionActive

Test the program, and you will see when an item is selected the background changes to MyBlue.

Step 8: Give visual feedback

I also want to give visual feedback to my user that indicates if a result is good or not. We will add a progressbar to the listboxitem template and bind the value property to the result. After that we will break the template of the progressbar and use our colors.

- Open the template of the listboxitem
- Select a progressbar from the toolbox and add it to the template
- Change the foreground from green to MyGray
- Set the minimum to 0 and the maximum to 20
- Click the gray square next to the value property
- Choose data binding
- Select the tab element property
- On the left side, select the textbox which contains the result
- On the right side, select the Text property
- Click Finish


Create Data Binding

Data Field | Element Property | Explicit Data Context

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

Properties
- ActualHeight : (Double)
- ActualWidth : (Double)
- ExtentHeight : (Double)
- ExtentWidth : (Double)
- FontSize : (Double)
- Height : (Double)
- HorizontalOffset : (Double)
- MaxHeight : (Double)
- MaxWidth : (Double)
- MinHeight : (Double)
- MinWidth : (Double)
- Name : (String)
- Opacity : (Double)
- SelectedText : (String)
- Text : (String)
- VerticalOffset : (Double)
- ViewportHeight : (Double)
- ViewportWidth : (Double)
- Width : (Double)

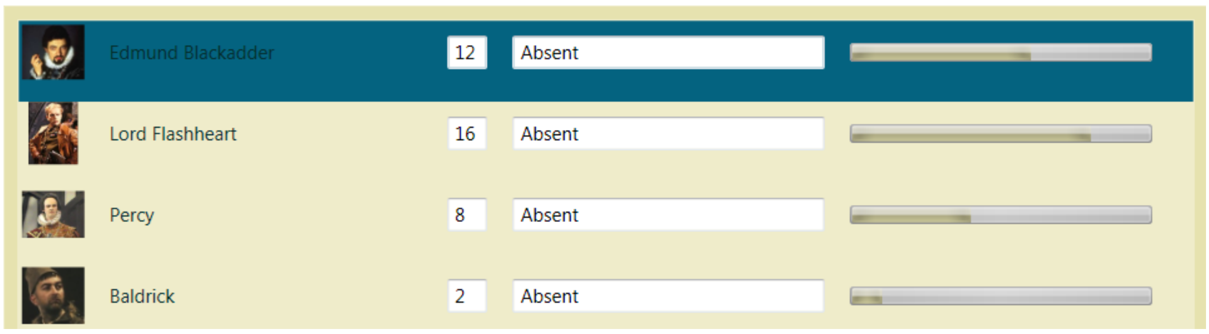
Show: Matching Types Only

☐ Use a custom path expression: Text

Finish | Cancel

- Right click the progressbar and choose Edit Control Parts (Template) → Edit template
- Choose a name for the template and save it in the application (reuse it on other progress bars)
- Select the Part_Indicator which contains a grid foreground
- Open the grid foreground and change the backgroundcolor for the indication and animation rectangles.

Test the application and you will see a progressbar in your own colors.



Edmund Blackadder	12	Absent
Lord Flashheart	16	Absent
Percy	8	Absent
Baldrick	2	Absent

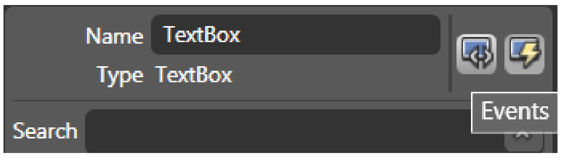
Step 9: Change the behavior of the listbox

In this final step, we will change the behavior of the listbox in two ways:

- If I press the TAB key in the last textbox (comment) then the next listboxitem must be selected
- If I click in a textbox, this must become the selected item.

We need to call different events on the textbox for this:

- Go back to the template of the listboxitem
- Select the stackpanel and search the Tag property
- Click the square near the Tag property and choose Data Binding…
- Select the tabpage Explicit Data Context, choose the element ResultList and click finish.
- Select the textbox with the result
- Click the events button in the properties panel


Name TextBox
Type TextBox
Search

- Choose the GotFocus event and give a name to it (for example: Result_GotFocus)
- Visual Studio will open and you can insert this code:

```
private void Result_GotFocus(object sender, RoutedEventArgs e)
        {
            giveFocus(sender);
        }

        private void giveFocus(object sender)
        {
            TextBox txt = (TextBox)sender;
            StackPanel stp = (StackPanel)txt.Parent;

            lstResults.SelectedItem = (StudentResult)stp.Tag;
        }
```

- Do the same for the textbox with the comment.
- Go back to Expression Blend and select the keydown event of the textbox with the comment.
- Visual Studio will open again and you can insert this code:

```
// if the TAB is pressed, select the next element
if (e.Key == System.Windows.Input.Key.Tab)
{
   // only if not the last element is selected
   if (lstResults.SelectedIndex + 1 < lstResults.Items.Count)
   {
         lstResults.SelectedIndex += 1;
   }
   else
   {
         // else select the first element
         lstResults.SelectedIndex = 0;
   }
}
```

Step 10: Detect changes in the data

You can also detect changes in the results and write them back to your database. I describe this process in my tutorial on the ObservableCollection.