Lab 1

A Windows Application with Two Controls

Introduction

In this lab you will implement the **TwoControls** example program from scratch. This example will illustrate in detail the steps needed to create a new WPF application using Visual Studio 2010, and you will get practice with all the fundamental concepts of WPF that we've covered in this chapter.

Suggested Time: 30 minutes

Root Directory: **OIC\WpfCs**

Directories:	**Labs\Lab1**	(do your work here)
	Chap01\TwoControls\Step1	(answer to Part 1)
	Chap01\TwoControls\Step2	(answer to Part 2)

Part 1. Create a WPF Application with a StackPanel

In Part 1 you will use Visual Studio to create a WPF application. You will go on to create a StackPanel that has as children a TextBox and a Button. This first version does not provide an event handler for the button. Also, it does not handle sizing very well!

1. Use Visual Studio to create a new WPF application **TwoControls** in the Lab1 folder.

2. In Solution Explorer, delete the files **App.xaml** and **MainWindow.xaml**.

3. Add a new code file **Program.cs** to your project.

4. In **Program.cs** enter the following code, which does the minimum of creating Application and Window objects.

```
using System;
using System.Windows;
using System.Windows.Controls;

namespace TwoControls
{
   class TwoControls : Window
   {
      [STAThread]
      static void Main(string[] args)
      {
         Application app = new Application();
         app.Run(new TwoControls());
      }
```

```
      public TwoControls()
      {
      }
   }
}
```

5. Build and run. You should get a clean compile. You should see a main window, which has no title and an empty client area.

6. Add the following code to the **TwoControls** constructor.

```
      public TwoControls()
      {
         Title = "Two Controls Demo";
         Width = 288;
      }
```

7. Build and run. Now you should see a title and the width as specified.

8. Now we are going to set the Content of the main window to a new StackPanel that we create. To be able to visually see the StackPanel, we will paint the background with a beige brush, and we'll make the Margin of the StackPanel 10 device-independent pixels.

```
      public TwoControls()
      {
         Title = "Two Controls Demo";
         Width = 288;
         const int MARGINSIZE = 10;

         StackPanel panel = new StackPanel();
         Content = panel;

         panel.Background = Brushes.Beige;
         panel.Margin = new Thickness(MARGINSIZE);
      }
```

9. Build. You'll get a compiler error because you need a new namespace for the **Brushes** class.

10. Bring in the **System.Windows.Media** namespace. Now you should get a clean build. Run your application. You should see the StackPanel displayed as solid beige, with a small margin.

11. Next we will add a TextBox as a child of the panel. Since we will be referencing the TextBox in an event-handler method as well as the constructor, define a private data member **txtName** of type **TextBox**.

```
      private TextBox txtName;
```

12. Provide the following code to initialize **txtName** and add it as a child to the panel.

```
         txtName = new TextBox();
```

```
txtName.FontSize = 16;
txtName.HorizontalAlignment = HorizontalAlignment.Center;
txtName.Width = Width / 2;
panel.Children.Add(txtName);
```

13. Build and run. Now you should see the TextBox displayed, centered, at the top of the panel.

14. Next, add code to initialize a Button and add it as a child to the panel.

```
Button btnGreet = new Button();
btnGreet.Content = "Say Hello";
btnGreet.FontSize = 16;
btnGreet.HorizontalAlignment = HorizontalAlignment.Center;
panel.Children.Add(btnGreet);
```

15. Build and run. You should now see the two controls in the panel. You are now at Step1.

Part 2. Event Handling and Layout

In Part 2 you will handle the Click event of the button. You will also provide better layout of the two controls.

1. First, we'll handle the Click event for the button. Provide this code to add a handler for the Click event.

```
btnGreet.Click += ButtonOnClick;
```

2. Provide this code for the handler, displaying a greeting to the person whose name is entered in the text box.

```
void ButtonOnClick(object sender, RoutedEventArgs args)
{
   MessageBox.Show("Hello, " + txtName.Text, "Greeting");
}
```

3. Build and run. The program now has its functionality, but the layout needs improving.

4. Provide the following code to size the height of the window to the size of its content.

```
SizeToContent = SizeToContent.Height;
```

5. Build and run. Now the vertical sizing of the window is better, but the controls are jammed up against each other.

6. To achieve a more attractive layout, provide the following statements to specify a margin around the text box and the button. You have a reasonable layout (Step2).

```
txtName.Margin = new Thickness(MARGINSIZE);
...
btnGreet.Margin = new Thickness(MARGINSIZE);
```