Hands-On Lab

Silverlight Firestarter

Working with Panels, XAML, and Controls

**Contents**

[Lab 3: Working with Panels, XAML and Controls 3](#_Toc276638888)

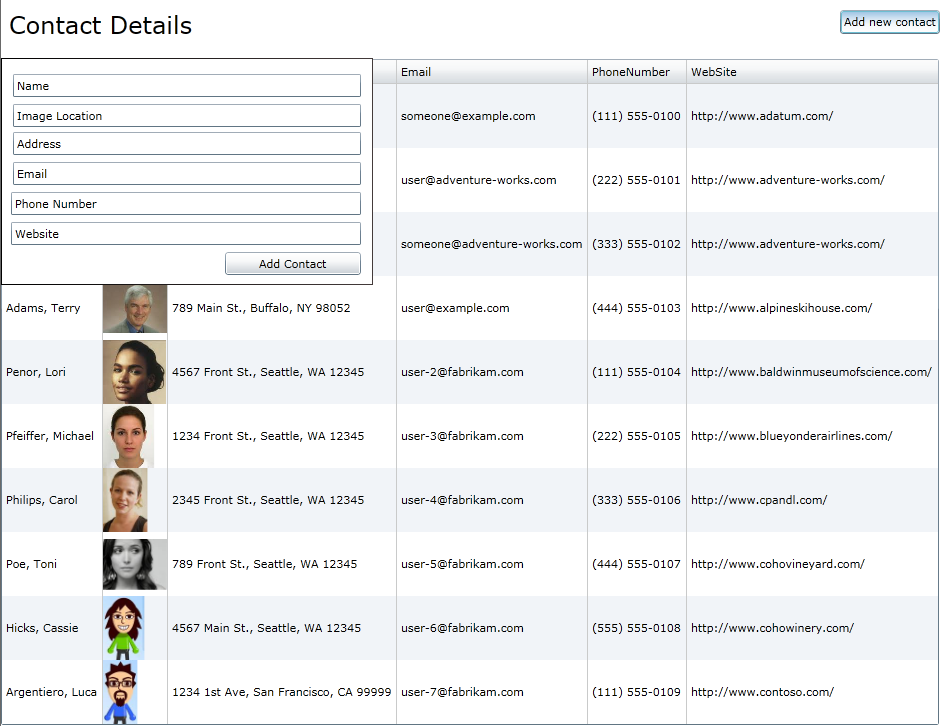
[Exercise 1: Creating a Data-Driven Interface 5](#_Toc276638889)

[Exercise 2: Creating a Form Entry to Edit Interface Details 9](#_Toc276638890)

[Exercise 3: Add a Menu using the StackPanel 18](#_Toc276638891)

[Summary 20](#_Toc276638892)

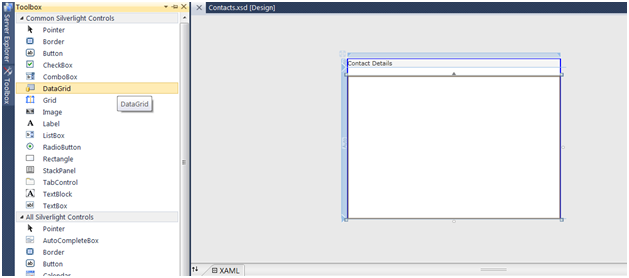
Lab 3: Working with Panels, XAML and Controls

* 1. This lab is designed to show how to work with Xaml, panels, controls, and Visual Studio’s Cider editor. You will learn the foundation of a Silverlight application’s UI.
  2. In the lab you'll create an application to manage contacts. The application will show the data in a grid, and you’ll be able to add a new contact with a data entry form. Along the way you'll learn how to add controls with Visual Studio, work with the Cider editor, as well as how effectively use panels.
  3. You'll start by creating a screen to view all your contacts, with the DataGrid control. Next, you’ll add a new UserControl that will be the interface to add a contact. Finally, you’ll use the StackPanel to hold the contents of a menu. The Silverlight application that you'll create is shown next:
     1. 
  4. **You Will Benefit from this Lab if:**
  + You are new to Silverlight
  + You are in need of a basic primer that’s business focused
  + You want to get a better understanding of how a Silverlight project is structured
  1. **You Will Learn:**
  + How to use the Visual Studio 2010 Silverlight Designer, Cider
  + XAML, Panels, and Controls
  + The difference between a Grid and a StackPanel
  + How to add controls
  + XML Namespaces
  + How to create a UserControl
  1. **Business Requirements for the Silverlight application include:**
  + Create a data driven interface with a DataGrid
  + Handle user interface events
  + Use XAML to create a data entry form
  + Handle the hiding and showing of the data input screen
  + Used the StackPanel to encapsulate the applications menu

Exercise 1: Creating a Data-Driven Interface

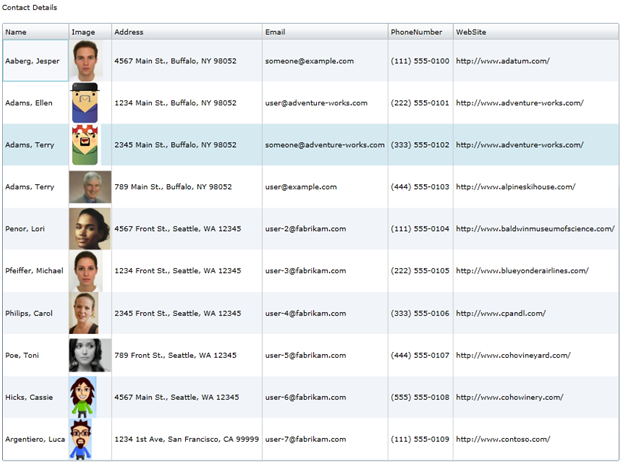
* 1. **Estimated Time: 45 minutes**
  2. Working with XAML is very similar to working with HTML, the main advantage being that you don’t have to worry about the pesky cross browser issues. Like HTML, XAML is XML. There are nodes, which define controls and attributes which are properties of the control. This exercise shows how to work with XAML, how to work with controls by adding a DataGrid, and how to work with the visual editor, known as the Cider editor, in Visual Studio.
  3. **Data-Driven Interface**
  4. Open the starter solution, and go to **MainPage.xaml**
  5. Define rows in the Grid, **LayoutRoot**,by using the**<Grid.RowDefinitions>** tag. Inside the tag, place a **<RowDefinition/>**for each intended row. The resulting **Layout**root should look like this:
     1. XAML
     2. <Grid x:Name="LayoutRoot" Background="White">
     3. <Grid.RowDefinitions>
     4. <RowDefinition Height="Auto"/>
     5. <RowDefinition Height="Auto"/>
     6. </Grid.RowDefinitions>

</Grid>

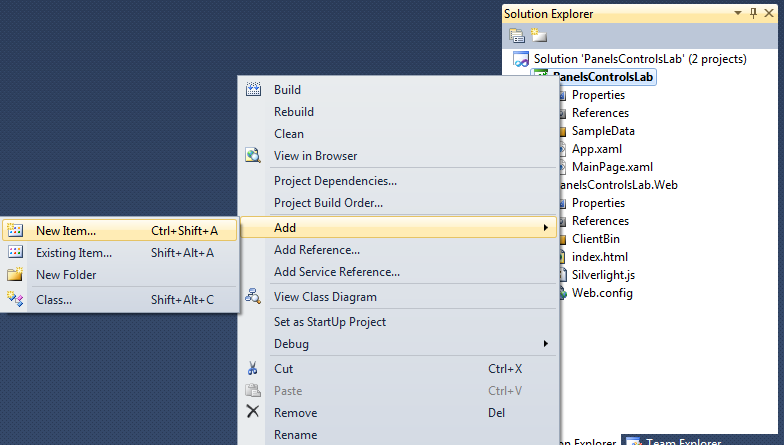
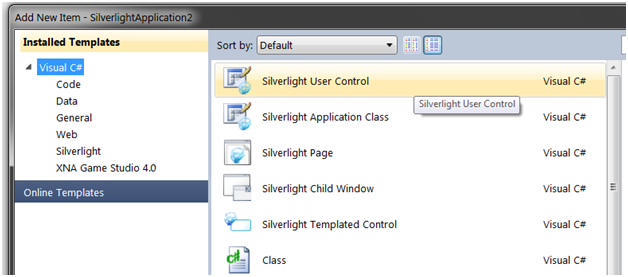
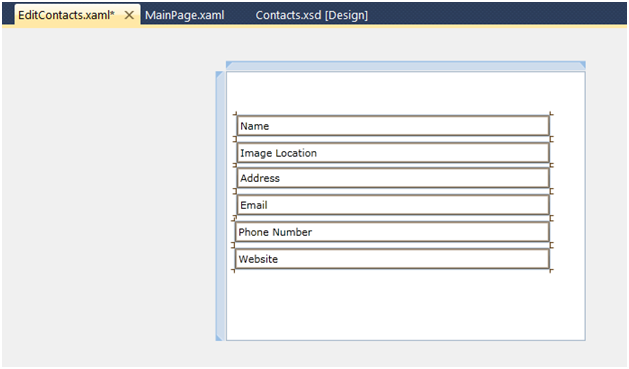
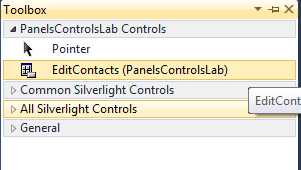
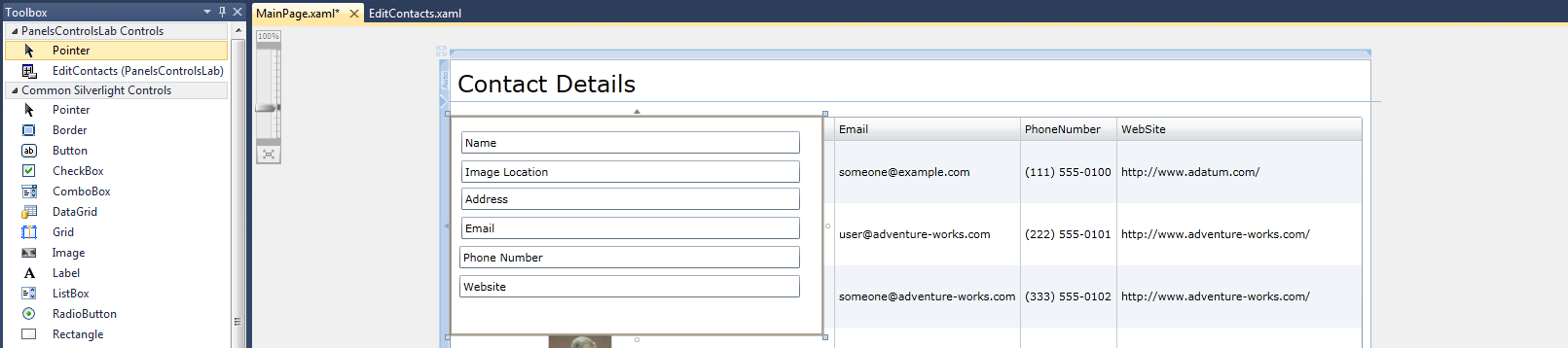
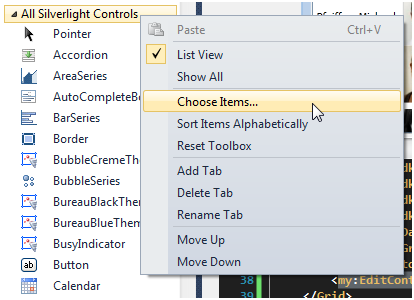
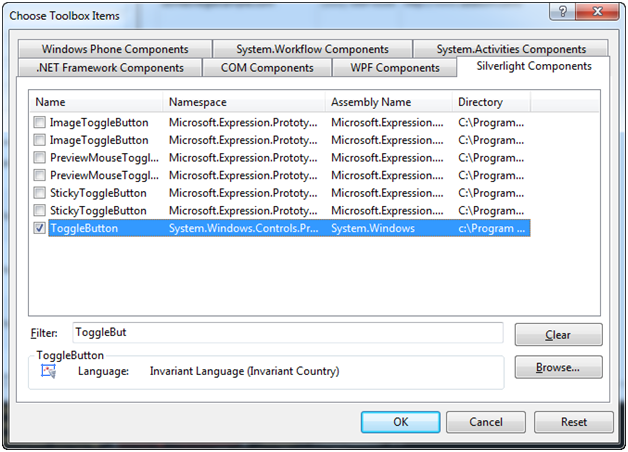
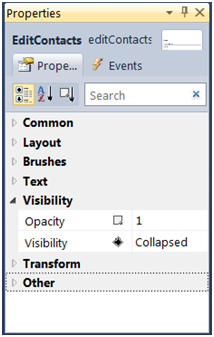
* + 1. **Note**: It’s important to note that you won’t place any tags specifying the beginnings and ends of rows inside the grid, itself. Instead, row assignment is specified as an attribute inside each control tag.
  1. Add a **TextBlock** control to serve as the application title. The row placement is specified using the **Grid.Row** attribute inside the **TextBlock** tag, as seen in the following Xaml.
     1. XAML
     2. <Grid x:Name="LayoutRoot" Background="White">
     3. <Grid.RowDefinitions>
     4. <RowDefinition Height="Auto"/>
     5. <RowDefinition Height="Auto"/>
     6. </Grid.RowDefinitions>
     7. <TextBlock Height="Auto" Width="Auto" Grid.Row="0">
     8. Contact Details
     9. </TextBlock>
     10. </Grid>
     11. **Note:** Rows are indexed from the top of the panel to the bottom, beginning with zero.
  2. Next, open the Toolbox on the left hand side of the screen and locate the **DataGrid** control. Drag and drop the **DataGrid** onto the design view below the header.
     1. 
     2. Figure 1
     3. DataGird in the toolbox
     4. Take a look at the XAML generated by Visual Studio for the **DataGrid** conrol. The **‘sdk:’** at the beginning of the tag denotes an **xmlns**(XML Namespace)declaration. At the top of the page, you can see where this declaration was included with the others. This is a necessary addition if you want to use the **DataGrid**, as it is part of the Silverlight Toolkit and not part of the native runtime.
     5. XAML
     6. d:DesignHeight="566" d:DesignWidth="947" xmlns:sdk="http://schemas.microsoft.com/winfx/2006/xaml/presentation/sdk">
     7. <Grid x:Name="LayoutRoot" Background="White">
     8. <Grid.RowDefinitions>
     9. <RowDefinition Height="Auto"/>
     10. <RowDefinition Height="Auto"/>
     11. </Grid.RowDefinitions>
     12. <TextBlock Height="Auto" Width="Auto" Grid.Row="0">

Contact Details

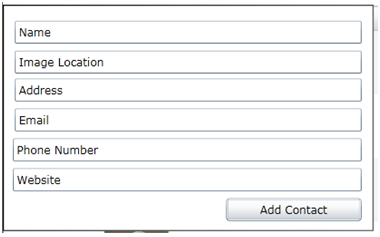
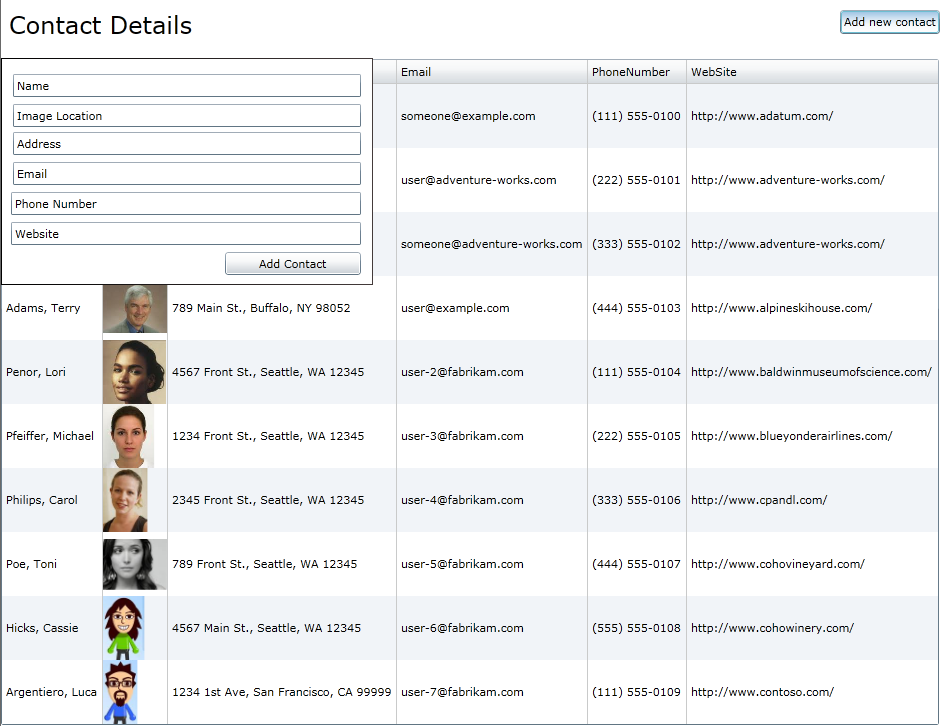
</TextBlock>

* + 1. <sdk:DataGrid AutoGenerateColumns="True" Grid.Row="1" Height="Auto"  
        Width="Auto" HorizontalAlignment="Left" Margin="0,0,16,0"   
        Name="dataGrid1" VerticalAlignment="Top"/>
    2. </Grid>
  1. Add data to the DataGrid. (For the purposes of this lab, the **DataGrid** is populated with a Sample Data Collection generated from Expression Blend. However, you can use your own web service, XML files, or C# class data to populate the **DataGrid**.) By setting **Height** and **Width** to ‘**Auto**’, the Grid dynamically resizes itself based on the size of the content. Here’s the finished XAML:
     1. XAML
     2. <UserControl.Resources>
     3. <DataTemplate x:Key="ImageTemplate">
     4. <StackPanel>
     5. <Image Source="{Binding Image}" HorizontalAlignment="Left"  
         Height="64" Width="64"/>
     6. </StackPanel>
     7. </DataTemplate>
     8. </UserControl.Resources>
     9. <Grid x:Name="LayoutRoot" Background="White" DataContext="{Binding Source={StaticResource Contacts}}">
     10. <Grid.RowDefinitions>
     11. <RowDefinition Height="Auto"/>
     12. <RowDefinition Height="Auto" />
     13. </Grid.RowDefinitions>
     14. <TextBlock Height="Auto" Width="Auto" Grid.Row="0" FontSize="24"  
          HorizontalAlignment="Left" VerticalAlignment="Top"  
          Margin="8,8,0,0">Contact Details</TextBlock>
     15. <sdk:DataGrid AutoGenerateColumns="False" Grid.Row="1" Height="Auto"   
          Width="Auto" HorizontalAlignment="Left" Margin="0,16,0,0"   
          x:Name="ContactsDataGrid" VerticalAlignment="Top"   
          ItemsSource="{Binding Collection}">
     16. <sdk:DataGrid.Columns>
     17. <sdk:DataGridTextColumn Binding="{Binding Name}" Header="Name"/>
     18. <sdk:DataGridTemplateColumn CellTemplate="{StaticResource   
          ImageTemplate}" Header="Image"/>
     19. <sdk:DataGridTextColumn Binding="{Binding Address}"   
          Header="Address"/>
     20. <sdk:DataGridTextColumn Binding="{Binding Email}" Header="Email"/>
     21. <sdk:DataGridTextColumn Binding="{Binding PhoneNumber}"   
          Header="PhoneNumber"/>
     22. <sdk:DataGridTextColumn Binding="{Binding WebSite}"   
          Header="WebSite"/>
     23. </sdk:DataGrid.Columns>
     24. </sdk:DataGrid>
     25. </Grid>
     26. And here’s what it looks like :
     27. 
     28. Figure 2
     29. The final result with some sample data

Exercise 2: Creating a Form Entry to Edit Interface Details

* + 1. Creating an UI for an entry form lends itself to the creation of a page. In ASP.NET you would create a WebForm. In a Silverlight project, a UserControl used to create a new page. In this exercise you will learn how to create a new UserControl and create a data entry form.
  1. **Creating a Form Entry**
  2. To add a new **UserControl**, right click on the solution name in the solution explorer and select ‘**add->New Item**’.
     1. 
     2. Figure 1
     3. Add new Item
  3. Select **Silverlight User Control** from the menu
     1. 
     2. Figure 2
     3. Add new item Menu
  4. Rename the Control to ‘**EditDetails.xaml**’ in the bottom of the Add New Item window, and click ‘Add’.
  5. Next create the form entries for each field in the **DataGrid** (Name, Image, Address, Email, Phone number, and Website). Using the Toolbox in Visual Studio, add 6 **TextBox** controls. Edit the ‘**Text**’ attribute for each block to add default text.
     1. XAML
     2. <TextBox Height="23" HorizontalAlignment="Left" Margin="12,16,0,0" Name="textBox1" VerticalAlignment="Top" Width="348" Text="Name" />
     3. <TextBox Height="23" HorizontalAlignment="Left" Margin="12,46,0,0" Name="textBox2" VerticalAlignment="Top" Width="348" Text="Image Location"/>
     4. <TextBox Height="23" HorizontalAlignment="Left" Margin="12,74,0,0" Name="textBox3" VerticalAlignment="Top" Width="348" Text="Address"/>
     5. <TextBox Height="23" HorizontalAlignment="Left" Margin="12,104,0,0" Name="textBox4" VerticalAlignment="Top" Width="348" Text="Email"/>
     6. <TextBox Height="23" HorizontalAlignment="Left" Margin="10,134,0,0" Name="textBox5" VerticalAlignment="Top" Width="350" Text="Phone Number"/>
     7. <TextBox Height="23" HorizontalAlignment="Left" Margin="10,164,0,0" Name="textBox6" VerticalAlignment="Top" Width="350" Text="Website"/>
     8. 
     9. Figure 3
     10. The resulting UI
  6. Place the new **UserControl** inside the **MainPage** View. Before you can do that, you must build the project in order the UserControl to show up in the Toolboox.
  7. Select **MainPage.xaml** and open the Toolbox. There should be a new entry at the top of your list.
     1. 
     2. Figure 4
     3. The EditContacts User Control in the toolbox
     4. **Note:** You must have a page other than **EditContacts.xaml** as your currently open document for EditContacts to appear in the toolbox.
  8. Much like adding the DataGrid, add the EditContacts UserControl by dragging and dropping onto the design view.
  9. Look at the Xaml and notice there is a new xml namespace, **xmlns:my**. This represents the current project, and lets the runtime know where to find the control.
     1. XAML
     2. xmlns:my="clr-namespace:PanelsControlsLab"
     3. 
     4. Figure 5
     5. The resulting UI
  10. At the top of the page, add a button to enable the user to open up the Add Contact interface. Drag and Drop a **ToggleButton** onto the design view. You can edit the **Content** attribute of the **ToggleButton** to say “Add new contact”.
      1. **Note:** If you don’t see the ToggleButton in the Toolbox, add it by right clicking the Toolbox and select Choose Items from the context menu.
      2. 
      3. Figure 6
      4. Right click the toolbox and select Choose Items…
      5. **Note:** From the Choose Toolbox Items dialog box, Find and select the **ToggleButton**. Now the **ToggleButton** is added to the Toolbox, find it and click and add it to your UI.
      6. 
      7. Figure 7
      8. Find ToggleButton under Silverlight Components
  11. The EditContacts UserControl should only be visible when the ToggleButton is checked, make sure this window doesn’t open by default. Right click on the EditDetails control and click properties to open the properties window. By default, the Properties panel opens in the bottom right and corner of the screen.
      1. 
      2. Figure 8
      3. In this window you can edit the visual properties of whatever control you have selected at the moment. For now, you want the default visibility set to **Collapsed**.
  12. Next, wire up that **ToggleButton** to hide and show the **EditDetails** control. In order to set this property, add an event handler to the code-behind.
      1. **Note:**There’s a shortcut if you want to add an Event Handler. If you select “Events” in the properties window of the **CheckBox** control, you’ll see an event labeled **Checked**. Double-click that event to automatically generate the event handler for the **CheckBox**. Do the same once more for the **UnChecked** event.
  13. In the **AddContactButton\_Checked** handler, set the visibility of the **EditContacts** control to **System.Windows.Visibility.Visible**. In the **AddContactButton\_UnChecked** handler set it to **System.Windows.Visibility.Collapsed**. Your **MainPage.xaml.cs** or **MainPage.xaml.vb** should look like this:
      1. C#
      2. namespace PanelsControlsLab
      3. {
      4. public partial class MainPage : UserControl
      5. {
      6. public MainPage()
      7. {
      8. InitializeComponent();
      9. }
      10. private void AddContactButton\_Checked(object sender,   
           System.Windows.RoutedEventArgs e)
      11. {
      12. this.AddContactPanel.Visibility =   
           System.Windows.Visibility.Visible;
      13. }
      14. private void AddContactButton\_Unchecked(object sender,   
           RoutedEventArgs e)
      15. {
      16. this.AddContactPanel.Visibility =System.Windows.Visibility.Collapsed;
      17. }
      18. }
      19. }
      20. Visual Basic
      21. Namespace PanelsControlsLab
      22. Partial Public Class MainPage
      23. Inherits UserControl
      24. Public Sub New()
      25. InitializeComponent()
      26. End Sub

Private Sub AddContactButton\_Checked(ByVal sender As Object,ByVal e As System.Windows.RoutedEventArgs)

* + 1. Me.AddContactPanel.Visibility = System.Windows.Visibility.Visible
    2. End Sub
    3. Private Sub AddContactButton\_Unchecked(ByVal sender As Object, ByVal e As RoutedEventArgs)
    4. Me.AddContactPanel.Visibility = System.Windows.Visibility.Collapsed
    5. End Sub
    6. End Class
    7. End Namespace
  1. Next add a button that will add the data entered in these fields into our DataGrid. Open **EditContacts.xaml** and drag a button onto the bottom of the UserControl. Set the text of the button to **Add Contact**.
     1. **Note:** (A simple border around the edit contacts window has been added, and set the Height and Width of **editContacts1** in MainPage.Xaml to ‘Auto’. This was done entirely for aesthetic purposes, and is optional).
     2. 
     3. Figure 9
     4. The finished EditContacrs in the MainPage.xaml designer
     5. XAML
     6. <Button Content="Add Contact" Height="23" HorizontalAlignment="Left" Margin="224,194,0,0" Name="button1" VerticalAlignment="Top" Width="136" />
     7. <Border BorderThickness="1" Height="227" HorizontalAlignment="Left" Name="border1" VerticalAlignment="Top" Width="372">
     8. <Border.BorderBrush>
     9. <LinearGradientBrush EndPoint="1,0.5" StartPoint="0,0.5">
     10. <GradientStop Color="Black" Offset="0" />
     11. <GradientStop Color="#FF443B3B" Offset="1" />
     12. </LinearGradientBrush>
     13. </Border.BorderBrush>
     14. </Border>
  2. That’s it! Run your program to see the finished product. Here what it will look like:
     1. 
     2. Figure 10
     3. The completed program

Exercise 3: Add a Menu using the StackPanel

* 1. So far you’ve learned how to add controls, use the DataGrid, work with the Grid, and add event handlers. In this exercise you’ll take a look at another type of Panel, the StackPanel. The StackPanel is unique because it handles aligning the contained elements for you. By specifying the Orientation property. The can be organized either Vertically or Horizontally. In this case you are going to add two more menu items (edit contacts and help) next to the Add new contact button and align the StackPanel horizontally.
  2. **Menu**
  3. Add a **ToggleButton** for editing the contacts and a **Button** for Help. Place the items next to the Add new contacts button. At this point you have your menu. Notice the margins and HorizontalAlignment of each button is set. For small scale applications, this is okay. However, what happens when you want to add more menu items, or better yet, if you are dynamically generating the menu.
  4. Create a StackPanel and move the controls inside. The below Xaml shows the how your menu should look.
     1. XAML
     2. <StackPanel HorizontalAlignment="Right" Margin="0,8,8,11"   
        Orientation="Horizontal">

<ToggleButton x:Name="AddContactButton" Content="Add new contact" Checked="AddContactButton\_Checked" Unchecked="AddContactButton\_Unchecked" Height="24" Margin="0,0,5,0" />

<ToggleButton x:Name="EditContactButton" Content="Edit contact" Checked="AddContactButton\_Checked" Unchecked="AddContactButton\_Unchecked" Height="24" Margin="0,0,5,0" />

* + 1. <Button x:Name="HelpButton" Content="Help"/>
    2. </StackPanel>
  1. And this is how the menu will look in Visual Studio
     1. 
     2. Figure 1
     3. The menu in Visual Studio

Summary

* 1. In this exercise you learned how to create a data driven Silverlight interface from the ground up. You added a data entry form to the existing prorject and created a menu with the StackPanel. In this application you satisfied the following requirements:
  + Create a data driven interface with a DataGrid
  + Handle user interface events
  + Use XAML to create a data entry form
  + Handle the hiding and showing of the data input screen
  + Used the StackPanel to encapsulate the applications menu
  1. This lab is the foundation for creating a data driven Silverlight interface. Other labs will provide additional details about application design practices such as Model-View-ViewModel (MVVM), how to use Expression Blend, how to integrate services, and how to create desktop applications with Silverlight.