Hands-On Lab

Building a Data-Driven Master/Detail Business Form using Visual Studio 2010

Lab version: 1.0.0

Last updated: 12/10/2010

CONTENTS

Overview

In this lab you will learn all the necessary steps for creating and customizing a master-detail business form in WPF 4.0 using Visual Studio 2010 tools.

Objectives

Once you've completed this lab you will understand:

- How to use the Data Sources window with a WPF project, to create initial scaffolding for the data bindings of your application
- How to use the Data Sources window to "paint" a databinding on an existing WPF control
- How to create master-detail scaffolding with the Data Sources window
- How to customize the output of the data sources window to create custom visuals using the Databinding expression builder in the Visual Studio 2010 property browser
- How to use markup extension intellisense to create databinding expressions
- How to extract a common look into a resource using the extract resource feature of Visual Studio 2010
- How to apply a common look to a control using the resource picker feature of Visual Studio 2010

Scenario

The application being built in this lab is a simple read-only representation of the sort of application that might be used to manage and view entries in a real estate agent's office. The techniques used could easily be applied to nearly any data-driven client application scenario.

Prerequisites

You should have a basic familiarity with the following products or technologies before you begin this lab:

- Data-driven application UI development
- Windows Presentation Foundation

System Requirements

You must have the following items to complete this lab:

- Microsoft Visual Studio 2010
- .NET Framework 4

Setup

All the requisites for this lab are verified using the **Configuration Wizard**. To make sure that everything is correctly configured, follow these steps:

Note: To perform the setup steps you need to run the scripts in a command window with administrator privileges.

1. Run the **Configuration Wizard** for the Training Kit if you have not done it previously. To do this, run the **CheckDependencies.cmd** script located under the **Setup** folder of this lab. Install any pre-requisites that are missing (rescanning if necessary) and complete the wizard.

   Note: For convenience, much of the code you will be managing along this lab is available as Visual Studio code snippets. The **CheckDependencies.cmd** file launches the Visual Studio installer file that installs the code snippets.

Exercises

This Hands-On Lab is comprised by the following exercises:

- Creating the Application's Data Scaffolding
- Creating Master-Detail Scaffolding
- Creating and Using Resources

Estimated time to complete this lab: **45 minutes**.

Note: Each exercise is accompanied by an **End** folder containing the resulting solution you should obtain after completing the exercises. You can use this solution as a guide if you need additional help working through the exercises.

Note: Each exercise contains a Visual Basic and a C# version; Inside the **End/Begin** solution folder you will find two folders: **VB**, containing the Visual Basic version of the exercise, and **C#**, containing the C# version of it.

For More Information

See the following videos in which this application is built up:

"What's New for Microsoft Silverlight and Microsoft Windows Presentation Foundation (WPF) Developers in Microsoft Visual Studio 2010"

http://videos.visitmix.com/MIX09/T73M

"What's New in Windows Presentation Foundation (WPF) 4"

http://videos.visitmix.com/MIX09/T39F

Next Step

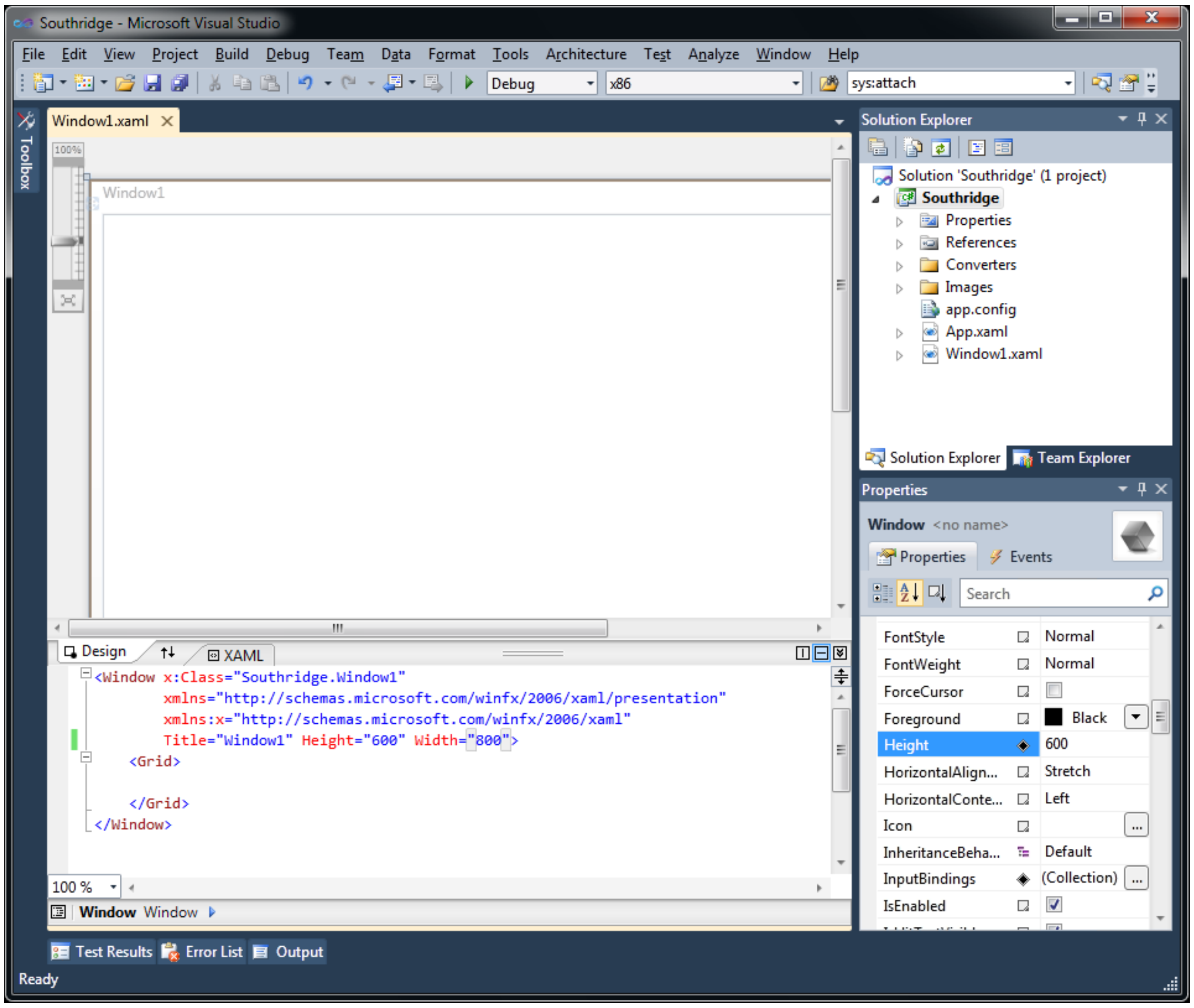
Exercise 1: Creating the application's data scaffolding, making basic customizations

Exercise 1: Creating the application's data scaffolding, making basic customizations

In this exercise, you will create the databinding scaffolding for your application, and display your first data. You will make some initial customizations of the data sources window's output.

Task 1 – Adding a Database to the Project Using Data Sources

1. Navigate to **Start** | **All Programs** | **Microsoft Visual Studio 2010** | **Microsoft Visual Studio 2010**.
2. Select the **File** | **Open** | **Project/Solution…** menu command. In the **Open Project** dialog, navigate to **\*Ex01-CreatingTheAppDataScaffolding*\begin\\ (choosing the folder that matches the language of your preference)** and select Southridge.sln file.
3. Open the **Window1.xaml** file by double-clicking on the file in the Solution Explorer.
4. Select the root window (be careful that you have the Window not the Grid selected – you can tell what is selected by checking the property browser window or the document outline window in Visual Studio). Use the Property Browser to set the **Width** of the window to 800 and the **Height** to 600

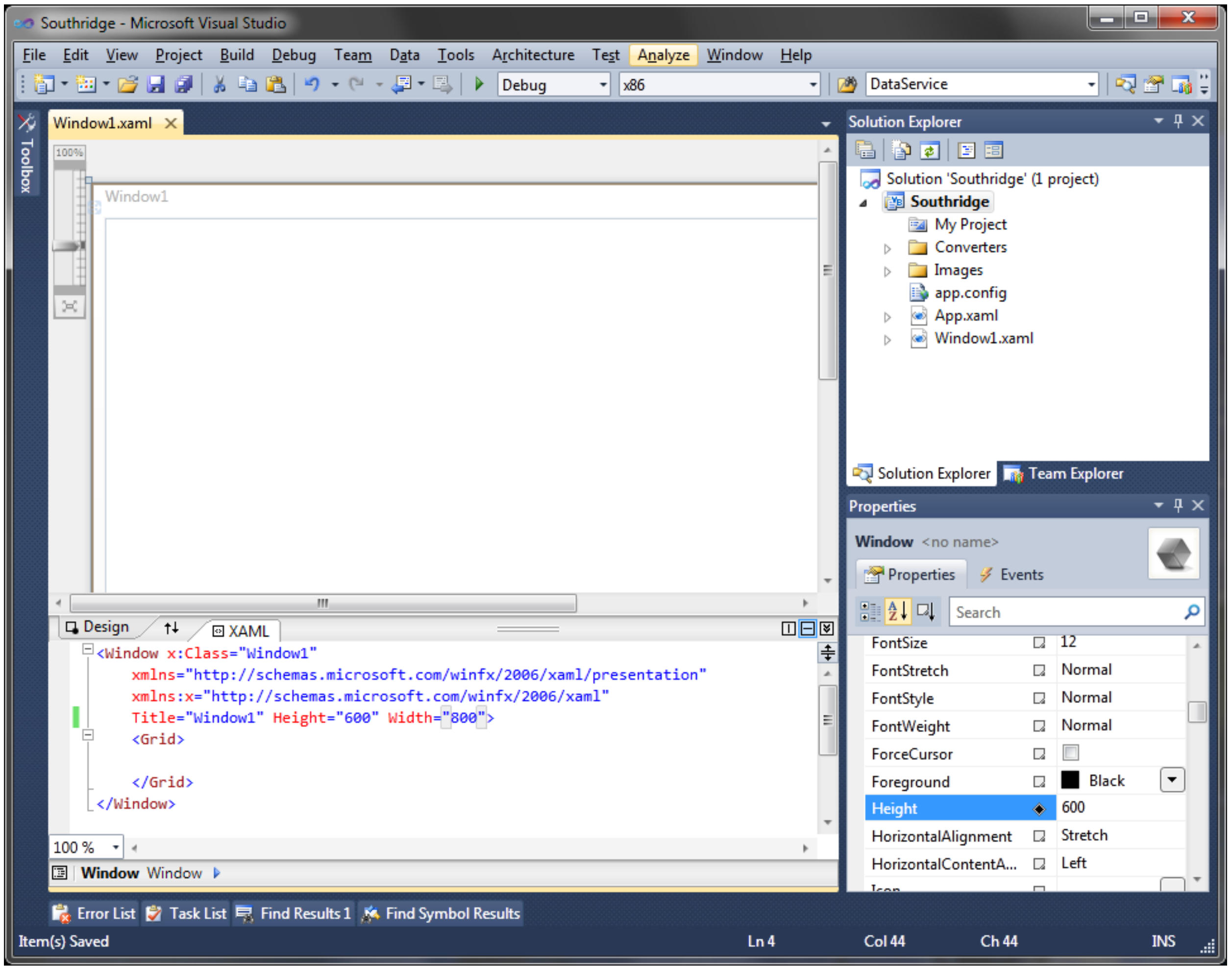


```
<Window x:Class="Southridge.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Window1" Height="600" Width="800">
    <Grid>

    </Grid>
</Window>
```

FontStyle	Normal
FontWeight	Normal
ForceCursor	
Foreground	Black
Height	600
HorizontalAlign...	Stretch
HorizontalConte...	Left
Icon	
InheritanceBeha...	Default
InputBindings	(Collection)
IsEnabled	✓

Figure 1

Updating Window1 Height and Width- C#



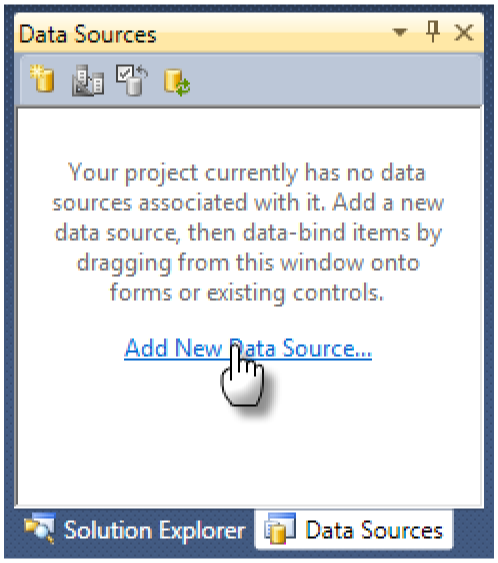
```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="600" Width="800">
    <Grid>

    </Grid>
</Window>
```

FontSize	12
FontStretch	Normal
FontStyle	Normal
FontWeight	Normal
ForceCursor	☐
Foreground	■ Black
Height	600
HorizontalAlignment	Stretch
HorizontalContentA...	Left

Figure 2

Updating Window1 Height and Width- Visual Basic

5. Open the Data Sources window by clicking **Data** | **Show Data Sources** menu option.
6. Create a new Data Source by clicking the **Add New Data Source…** hyperlink in the Data Sources window



Your project currently has no data sources associated with it. Add a new data source, then data-bind items by dragging from this window onto forms or existing controls.

Add New Data Source...

Figure 3
Data Sources

7. Choose the Data Source Type of **Database**. Click **Next**.
8. Choose **DataSet** as the Database Model and click **Next**.
9. Click **New Connection…** and choose the following options:
    a. **Data source**: Microsoft SQL Server Database File
    b. **Database file name**: \Assets\Southridge.mdf
    c. **Log on to server**: Use Windows Authentication
10. Click **Next**. Click **Yes** to copy the file to your local project.
11. Click **Next**, keeping the default connection string name.
12. On the Choose Your Database Objects screen:
    a. Under **Tables** | **Listings**, check **MLS**, **Title**, **Price** and **PrimaryPhoto**
    b. Check **Tables** | **Neighborhoods**
    c. Check **Tables** | **Viewings**
13. Click **Finish**.
14. You should now see the Data Sources window populated already with database schema. You will also see the related XSD strongly typed DataSet in the Solution Explorer.
15. Build the solution – you should see no build errors

Task 2 – Using Data Sources to Add Data-Bound Controls to the Window

1. In the Data Sources window, click the **Listings** dropdown and select **Details**.

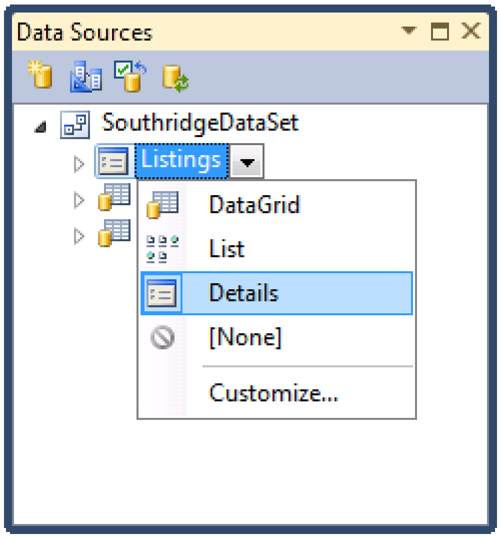
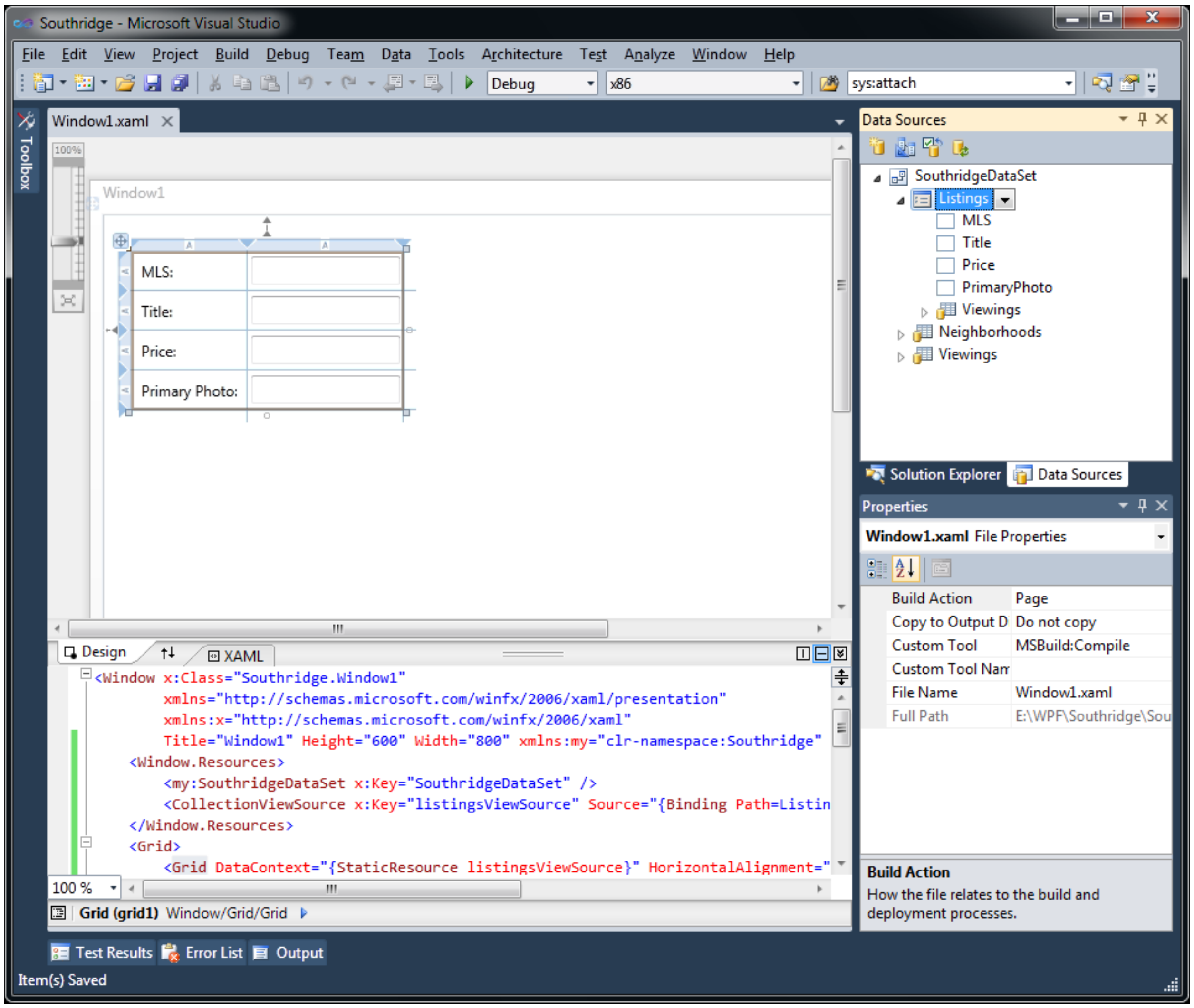


Figure 4
Listings Details

2. Drag and drop from the Listing dropdown onto Windows1
3. Note that the Data-Bindings and Controls are created automatically. Inspect the XAML that was created for you and observe its relationship to the SouthridgeDataSet data source.



MLS:
Title:
Price:
Primary Photo:

```
<Window x:Class="Southridge.Window1"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        Title="Window1" Height="600" Width="800" xmlns:my="clr-namespace:Southridge">
    <Window.Resources>
        <my:SouthridgeDataSet x:Key="SouthridgeDataSet" />
        <CollectionViewSource x:Key="listingsViewSource" Source="{Binding Path=Listin
    </Window.Resources>
    <Grid>
        <Grid DataContext="{StaticResource listingsViewSource}" HorizontalAlignment="
```

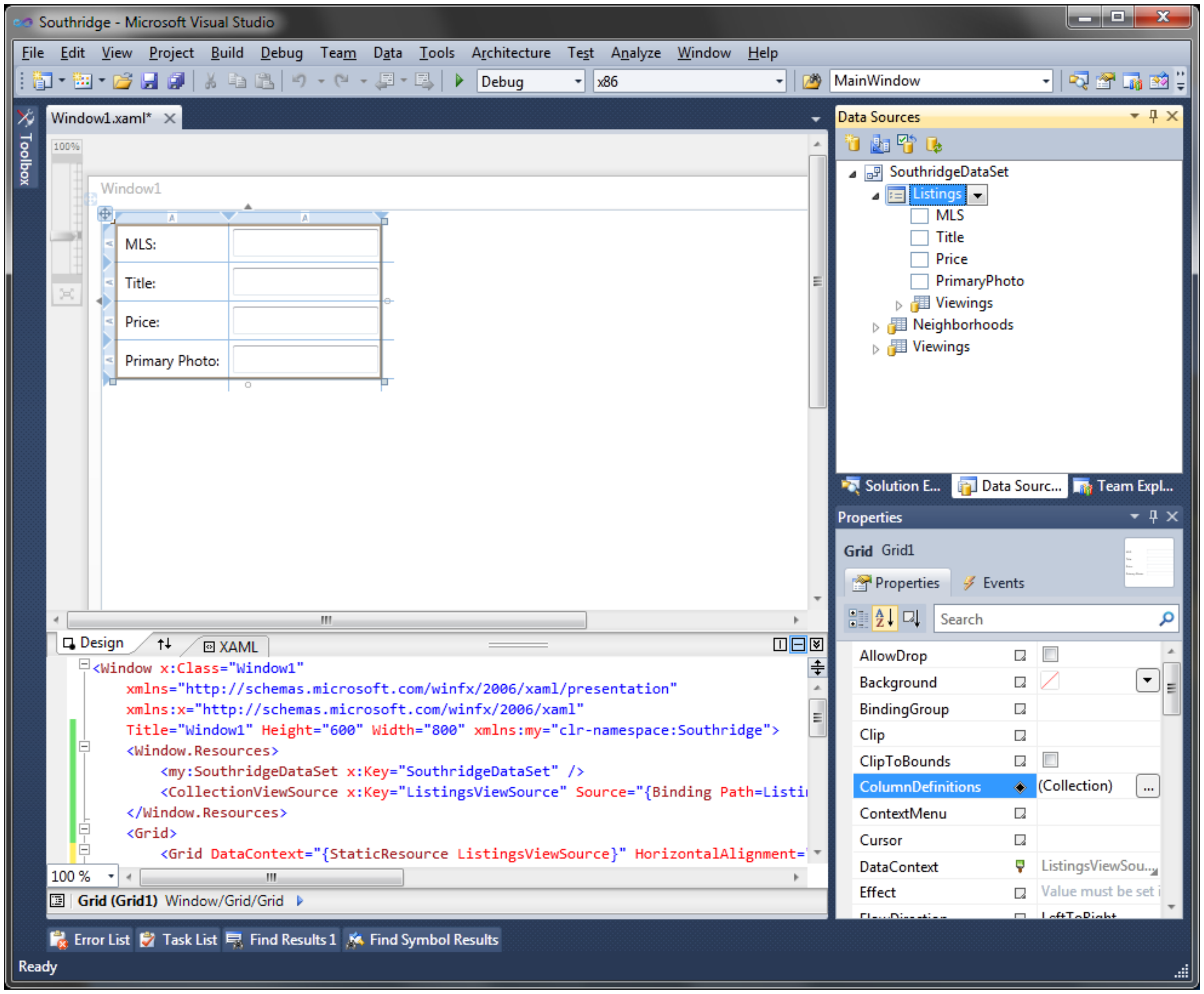
Window1.xaml File Properties

Build Action	Page
Copy to Output D	Do not copy
Custom Tool	MSBuild:Compile
Custom Tool Nam	
File Name	Window1.xaml
Full Path	E:\WPF\Southridge\Sou

Build Action
How the file relates to the build and deployment processes.

Figure 5

Listing Details in WPF Designer – C#



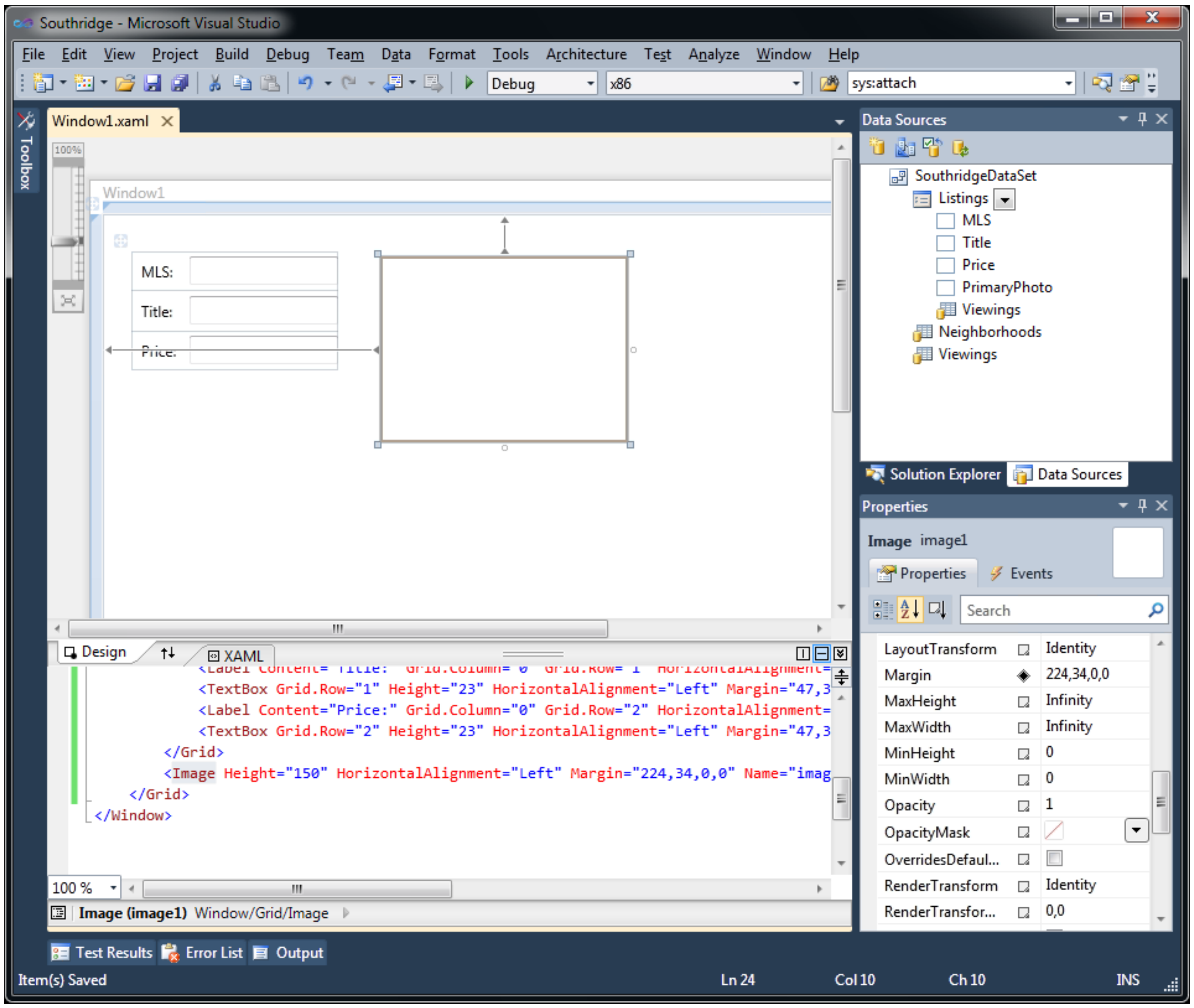
```
<Window x:Class="Window1"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="Window1" Height="600" Width="800" xmlns:my="clr-namespace:Southridge">
    <Window.Resources>
        <my:SouthridgeDataSet x:Key="SouthridgeDataSet" />
        <CollectionViewSource x:Key="ListingsViewSource" Source="{Binding Path=Listi
    </Window.Resources>
    <Grid>
        <Grid DataContext="{StaticResource ListingsViewSource}" HorizontalAlignment="
```

AllowDrop	
Background	
BindingGroup	
Clip	
ClipToBounds	
ColumnDefinitions	(Collection)
ContextMenu	
Cursor	
DataContext	ListingsViewSou...
Effect	Value must be set i

Figure 6

Listing Details in WPF Designer – Visual Basic

4. On the Design pane, delete the Primary Photo **Label** and **Textbox**.
5. Drag and drop and **Image** control from the **Toolbox** and place to the right of the details grid.
6. In the **Data Sources** window, expand the **Listings** table and then drag and drop the PrimaryPhoto field onto the **Image** control. This will “paint” the data binding information on to the control.



```
            <TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="47,3
            <Label Content="Price:" Grid.Column="0" Grid.Row="2" HorizontalAlignment=
            <TextBox Grid.Row="2" Height="23" HorizontalAlignment="Left" Margin="47,3
        </Grid>
        <Image Height="150" HorizontalAlignment="Left" Margin="224,34,0,0" Name="imag
    </Grid>
</Window>
```

LayoutTransform	Identity
Margin	224,34,0,0
MaxHeight	Infinity
MaxWidth	Infinity
MinHeight	0
MinWidth	0
Opacity	1
OpacityMask	
OverridesDefaul...	
RenderTransform	Identity
RenderTransfor...	0,0

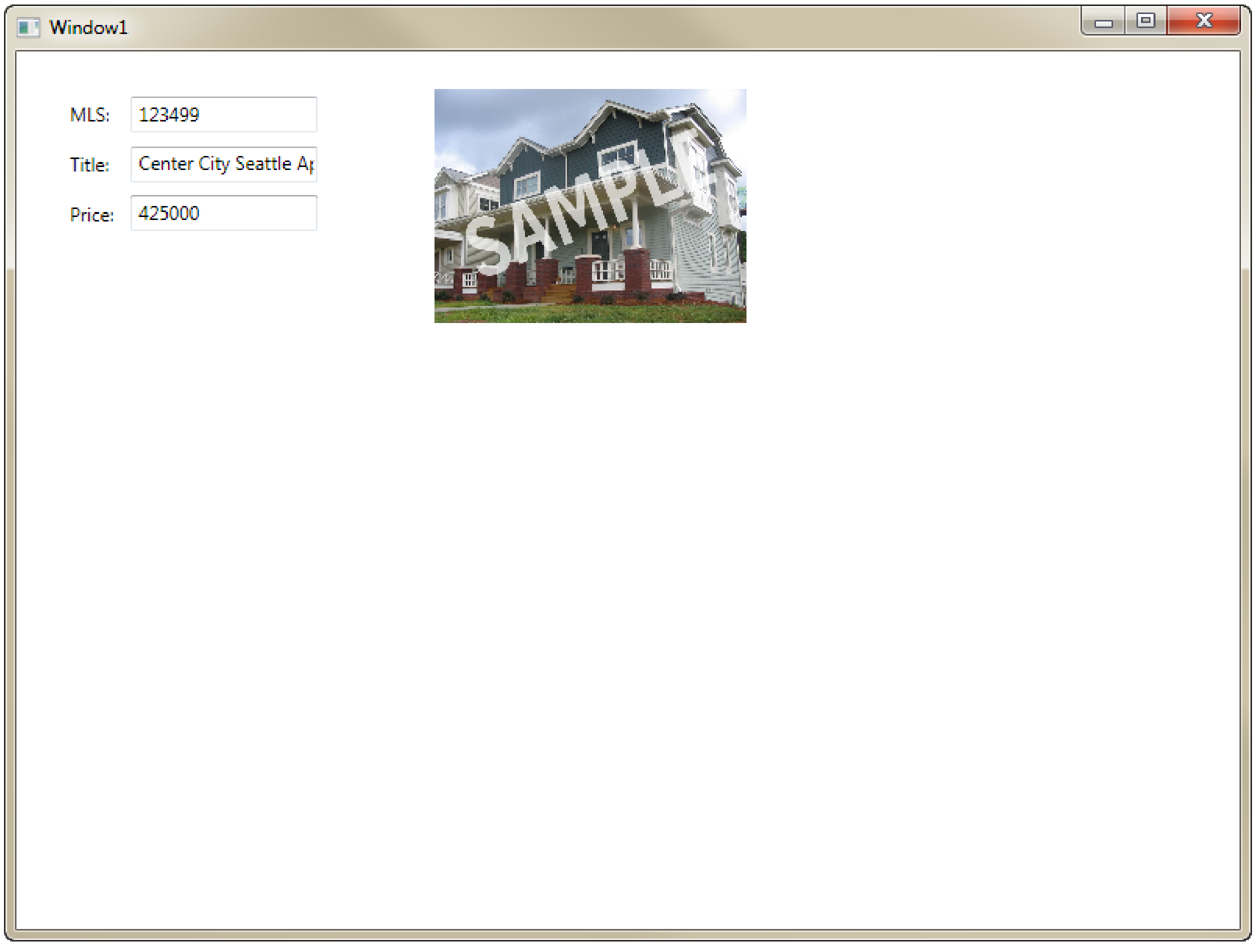
100 %

Image (image1) Window/Grid/Image

Figure 7

Adding an Image Control

7. Press Ctrl-F5 to run the application. You should see data appearing from the first row in the database, and a picture showing.



MLS: 123499
Title: Center City Seattle Ap
Price: 425000

Figure 8
Running the Application

Next Step

Exercise 2: Creating master-detail data scaffolding, making more detailed customizations

Exercise 2: Creating master-detail data scaffolding, making more detailed customizations

In this exercise, you will add some additional detail to the window. You will also provide a way for the user to navigate forward and backward through the listings. To help visualize the data a little better, you will also override one of the default templates and provide a visual indication if a listing is proceeding or not.

Task 1 – Adding a Listing Details Grid to the Window

1. For this exercise, you can continue to use the solution from Exercise 1 or open the begin solution located at ***\Ex02-CreatingMasterDetailDataScaffolding\begin\*** **(Choosing the folder that matches the language of your preference.)** You will need to build the solution (click **Build** | **Build Solution**) before the designer will show the WPF work done in Exercise 1.

2. If the Data Sources window is not show, click **Data** | **Show Data Sources**.

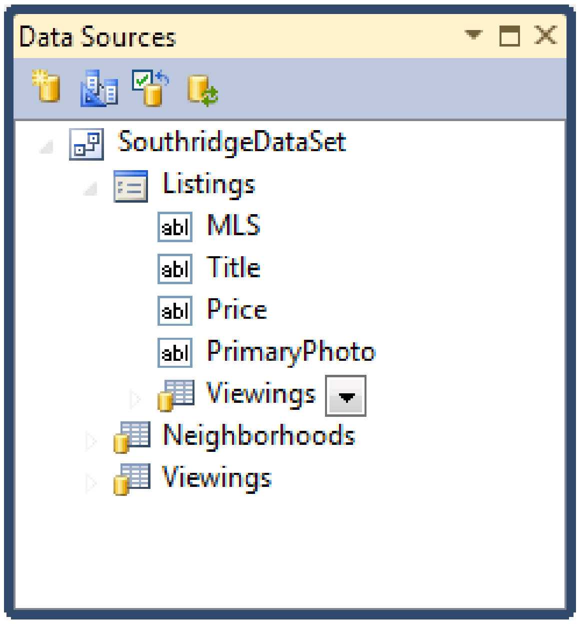
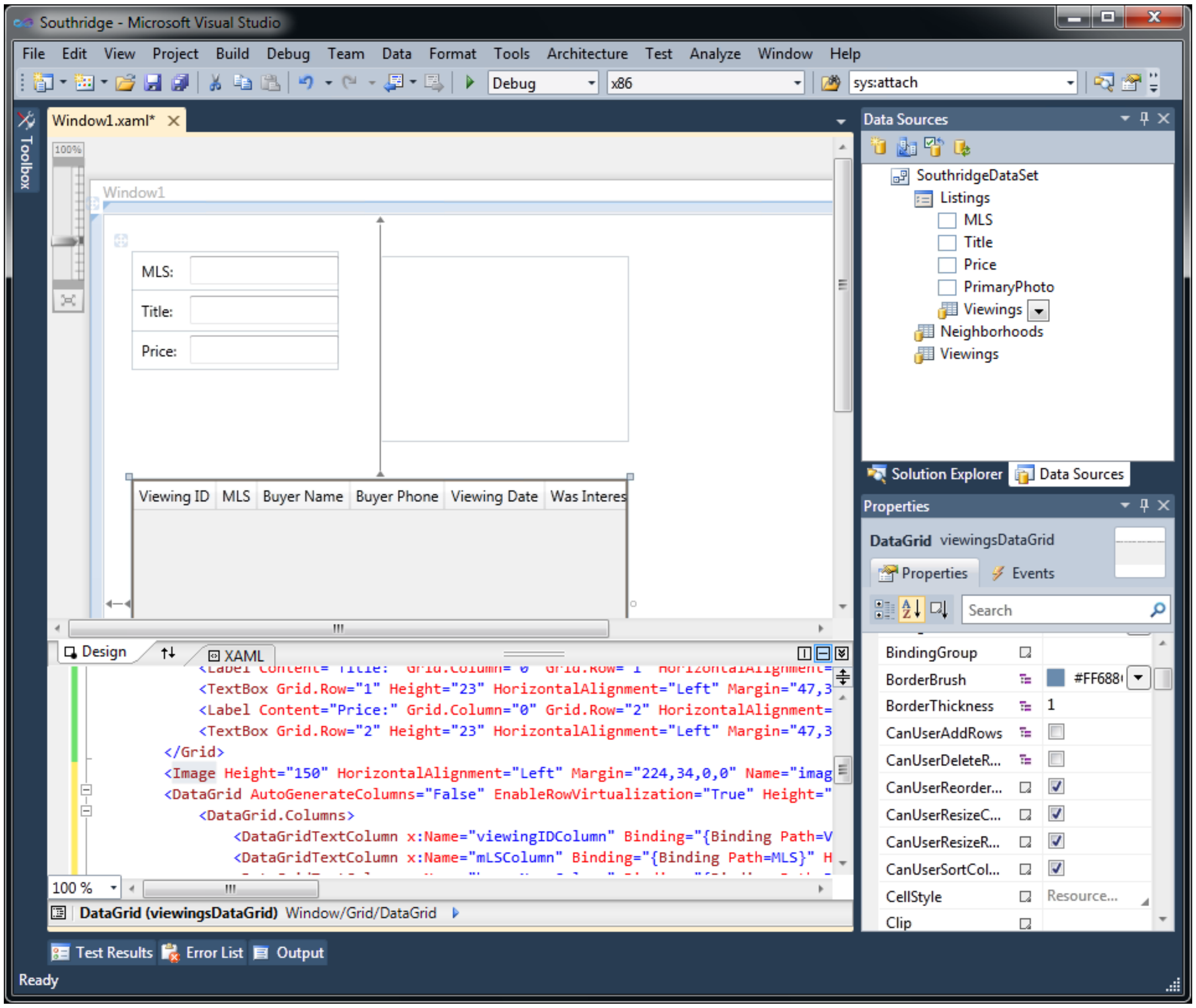


Figure 9

Viewings in Data Sources

3. Drag the **Viewings** Foreign Key table from under **Listings** in the Data Sources window onto the Window to make a master-details view. Note that the data sources window takes care of all the details of keeping the two views in sync, and creates a **DataGrid** with all your fields in different columns:



```
<Label Content="Title:" Grid.Column="0" Grid.Row="1" HorizontalAlignment=
<TextBox Grid.Row="1" Height="23" HorizontalAlignment="Left" Margin="47,3
<Label Content="Price:" Grid.Column="0" Grid.Row="2" HorizontalAlignment=
<TextBox Grid.Row="2" Height="23" HorizontalAlignment="Left" Margin="47,3
</Grid>
<Image Height="150" HorizontalAlignment="Left" Margin="224,34,0,0" Name="imag
<DataGrid AutoGenerateColumns="False" EnableRowVirtualization="True" Height="
    <DataGrid.Columns>
        <DataGridTextColumn x:Name="viewingIDColumn" Binding="{Binding Path=V
        <DataGridTextColumn x:Name="mLSColumn" Binding="{Binding Path=MLS}" H
```

DataGrid (viewingsDataGrid) Window/Grid/DataGrid

DataGrid viewingsDataGrid

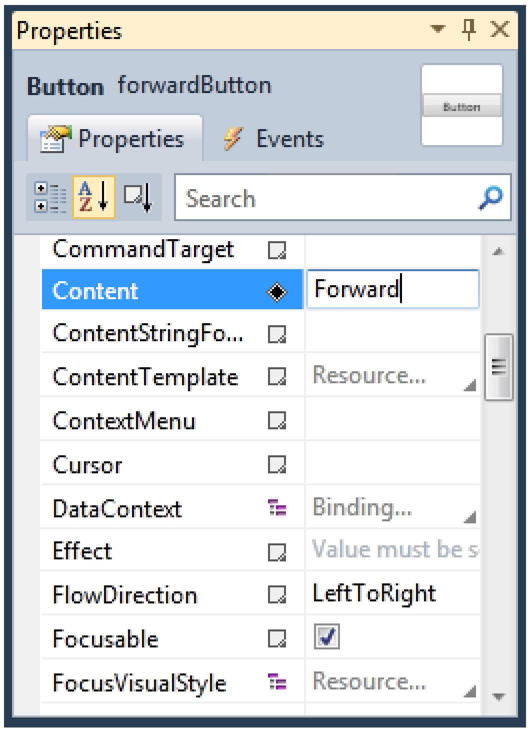
BindingGroup	
BorderBrush	#FF688
BorderThickness	1
CanUserAddRows	☐
CanUserDeleteR...	☐
CanUserReorder...	✓
CanUserResizeC...	✓
CanUserResizeR...	✓
CanUserSortCol...	✓
CellStyle	Resource...
Clip	

Figure 10

Adding the Viewings Into the Designer

Task 2 – Providing Forward and Back Buttons for Navigation

1. Drag 2 new **Buttons** onto the design surface from the toolbox
2. Set the **Name** of the first Button to *forwardButton*.
3. Set the **Content** property of the **forwardButton** to *Forward*.



Button forwardButton

CommandTarget	
Content	Forward
ContentStringFo...	
ContentTemplate	Resource...
ContextMenu	
Cursor	
DataContext	Binding...
Effect	Value must be s
FlowDirection	LeftToRight
Focusable	✓
FocusVisualStyle	Resource...

Figure 11

Forward Button Properties

4. Double-click the **forwardButton** on the designer surface. Visual Studio will switch to code view and add a click handler for this button.
5. Insert the following code into this new click handler.

(Code Snippet – *WPF MasterDetail Lab – Ex2 ForwardClickHandler C#*)

C#

```
private void forwardButton_Click(object sender, RoutedEventArgs e)
    {
    CollectionViewSource cvs = (this.FindResource("listingsViewSource"))
                               as CollectionViewSource;
    cvs.View.MoveCurrentToNext();
}
```

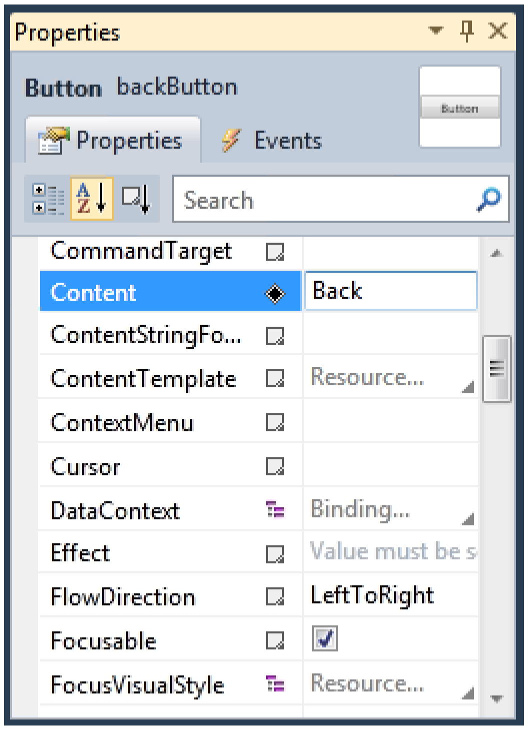
(Code Snippet – *WPF MasterDetail Lab – Ex2 ForwardClickHandler VB*)

Visual Basic

```
Private Sub forwardButton_Click(ByVal sender As System.Object, ByVal e As
System.Windows.RoutedEventArgs) Handles forwardButton.Click
    Dim cvs = TryCast(Me.FindResource("ListingsViewSource"),
CollectionViewSource)
    cvs.View.MoveCurrentToNext()
```

```
End Sub
```

6. Switch back to by right-clicking in the code editor and select **View Designer**.
7. Set the **Name** of the second Button to *backButton*.
8. Set the **Content** property of the **backButton** to *Back*.



CommandTarget	
Content	Back
ContentStringFo...	
ContentTemplate	Resource...
ContextMenu	
Cursor	
DataContext	Binding...
Effect	Value must be s
FlowDirection	LeftToRight
Focusable	✓
FocusVisualStyle	Resource...

Figure 12
Back Button Properties

9. Double-click the **backButton** on the designer surface. Visual Studio will switch to code view and add a click handler for this button.
10. Insert the following code into this new click handler.

(Code Snippet – *WPF MasterDetail Lab – Ex2 BackClickHandler C#*)

C#

```
private void backButton_Click(object sender, RoutedEventArgs e)
{
    CollectionViewSource cvs = (this.FindResource("listingsViewSource"))
                               as CollectionViewSource;
    cvs.View.MoveCurrentToPrevious();
}
```

(Code Snippet – *WPF MasterDetail Lab – Ex2 BackClickHandler VB*)

Visual Basic

```
Private Sub backButton_Click(ByVal sender As System.Object, ByVal e As
System.Windows.RoutedEventArgs) Handles backButton.Click
    Dim cvs = TryCast(Me.FindResource("ListingsViewSource"),
CollectionViewSource)
    cvs.View.MoveCurrentToPrevious()
End Sub
```

11. Click **Debug** | **Start without Debugging** (or press Ctrl-F5) to run the app and note that you can page back and forth through your data set, seeing the details for each item as you select it.

Task 3 – Providing a Visualization for the Data in the Listings DataGrid

In this task, you will make a more detailed customization of the DataGrid you just created. The goal is to change one of the columns from showing CheckBox controls to showing a simple Red/Green traffic light style rectangle as an indicator. To do this, you will use a **BooleanToBrushConverter** already created in the project, to convert the Boolean value in a column to a Red or Green brush.

1. Select the DataGrid control on the design surface by clicking on it. Right-click the design surface and choose Document Outline.

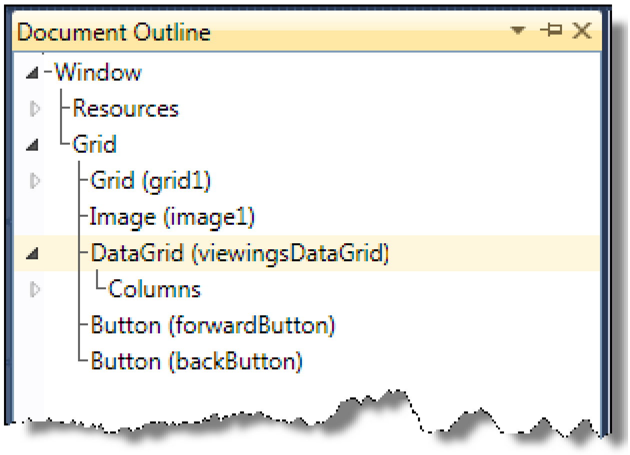


Figure 13
DataGrid in Document Outline

2. Expand the DataGrid node in the Document Outline, and find the 3rd-from-bottom Column inside the DataGrid. This column will be a DataGridCheckBoxColumn and you will notice that the XAML editor highlights the related code.

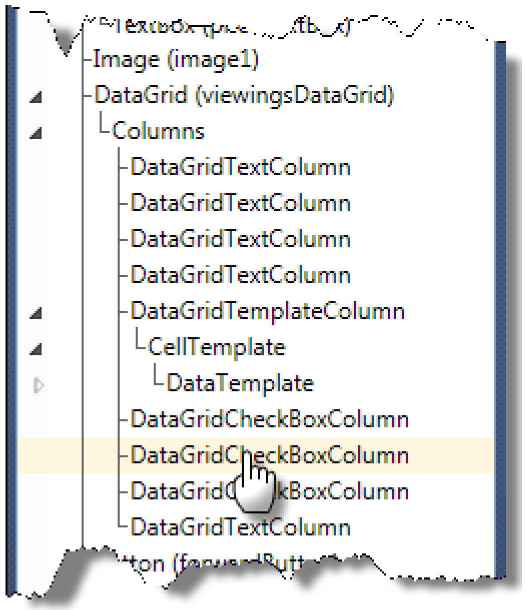


Image (image1)
DataGrid (viewingsDataGrid)
Columns
DataGridTextColumn
DataGridTextColumn
DataGridTextColumn
DataGridTextColumn
DataGridTemplateColumn
CellTemplate
DataTemplate
DataGridCheckBoxColumn
DataGridCheckBoxColumn
DataGridCheckBoxColumn
DataGridTextColumn

Figure 14

DataGridCheckBoxColumn in Document Outline

3. In the XAML editor, comment out the DataGridCheckBoxColumn and insert the following DataGridTemplateColumn.

XAML

```
<!-- <DataGridCheckBoxColumn Binding="{Binding Path=Proceeding}"
                Header="Proceeding" Width="SizeToHeader" /> -->
<DataGridTemplateColumn Header="Proceeding" Width="SizeToHeader">
    <DataGridTemplateColumn.CellTemplate>
        <DataTemplate>
             <Rectangle />
        </DataTemplate>
    </DataGridTemplateColumn.CellTemplate>
</DataGridTemplateColumn>
```

4. Place your cursor on the <Rectangle /> code and the Properties pane will show the properties for the Rectangle.
5. In the Properties window, use the Search field to find the Fill attribute.

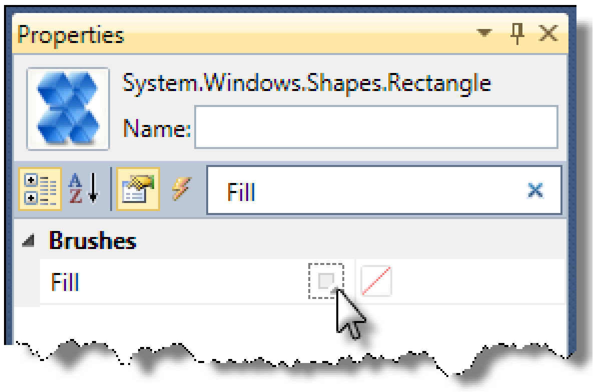


Figure 15
Rectangle Fill Property

6. Click the small black square located right next to the word Fill. Click **Apply Data Binding**.
7. In the Data Binding window, click the Converter pane and select **Southridge**, **BooleanToBrushConverter** and **Create New…**

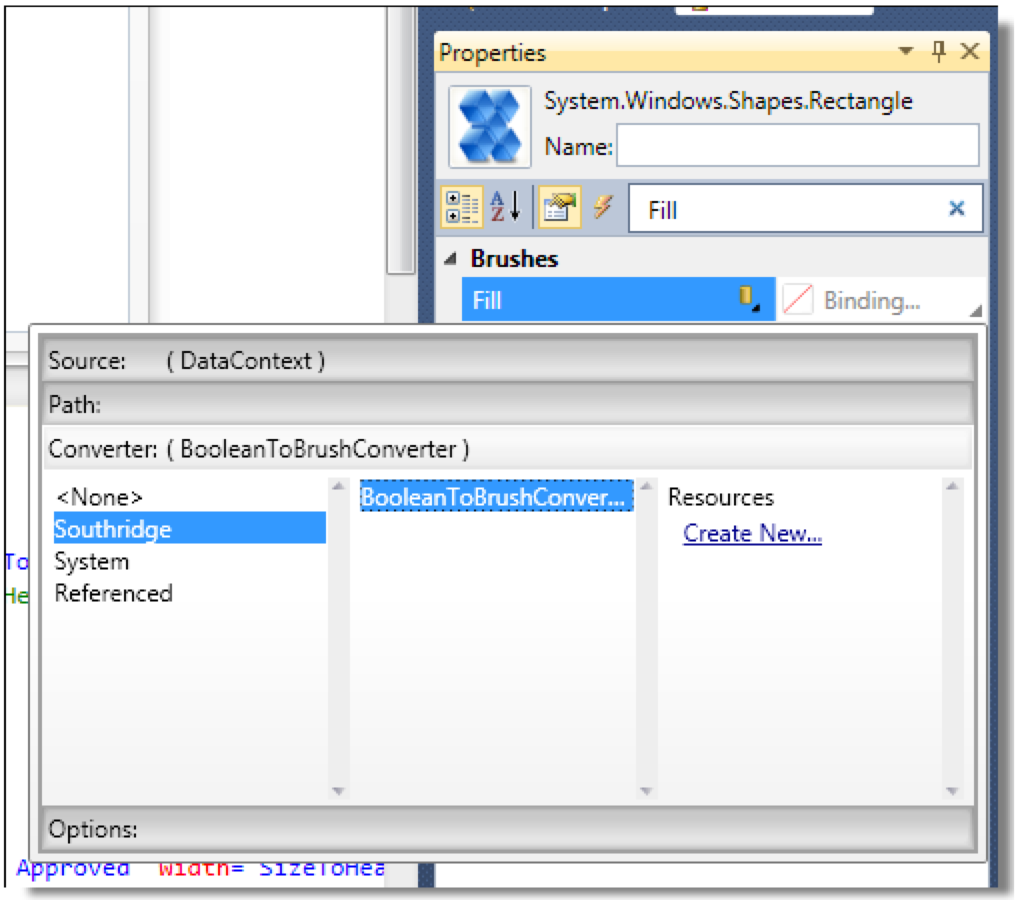


Figure 16
Adding BooleanToBrushConverter

8. Click **OK** in the Create Resource dialogue, keeping the default value for Key.
9. Type *Proceeding* in the **Parameter** field. This will tell the Converter which field to get the value from in the current Row.

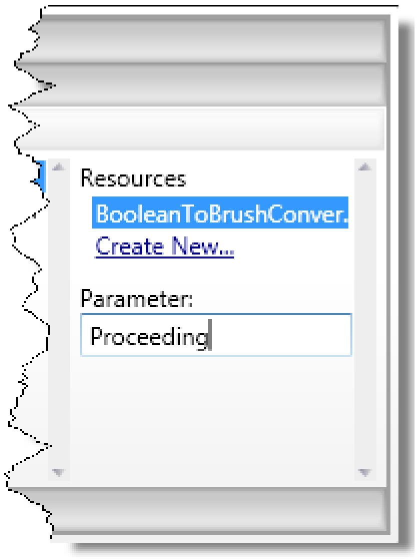
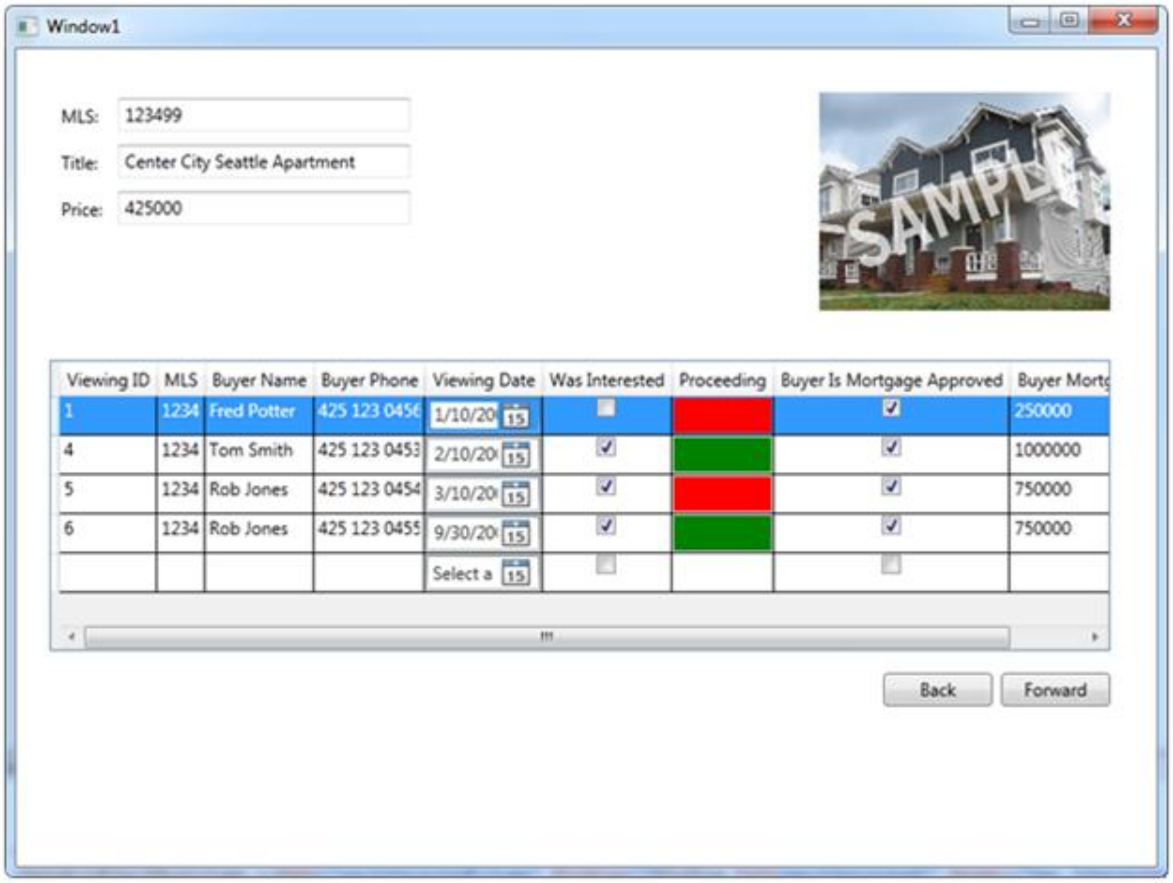


Figure 17
BooleanToBrushConverter Parameter

10. Click **Debug** | **Start without Debugging** (or press Ctrl-F5) and see your traffic lights in the proceeding column!



MLS: 123499

Title: Center City Seattle Apartment

Price: 425000

Viewing ID	MLS	Buyer Name	Buyer Phone	Viewing Date	Was Interested	Proceeding	Buyer Is Mortgage Approved	Buyer Mortg
1	1234	Fred Potter	425 123 0456	1/10/20		🟥	✓	250000
4	1234	Tom Smith	425 123 0453	2/10/20	✓	🟩	✓	1000000
5	1234	Rob Jones	425 123 0454	3/10/20	✓	🟥	✓	750000
6	1234	Rob Jones	425 123 0455	9/30/20	✓	🟩	✓	750000
				Select a				

Figure 18

Running the Application

Next Step

Exercise 3: Creating and using resources

Exercise 3: Creating and using resources

Task 1 – Creating a Resource from the Background Property of a Button

1. For this exercise, you can continue to use the solution from Exercise 2 or open the begin solution located at ***\Ex03-CreatedAndUsingResources\begin\*** **(Choosing the folder that matches the language of your preference.)** You will need to build the solution (click **Build** | **Build Solution**) before the designer will show the WPF work we did in Exercise 2.

2. Open Window1.xaml and select the **forwardButton** created in Exercise 2.

3. In the Properties window, use the Search tool to find the **Background** property**.**

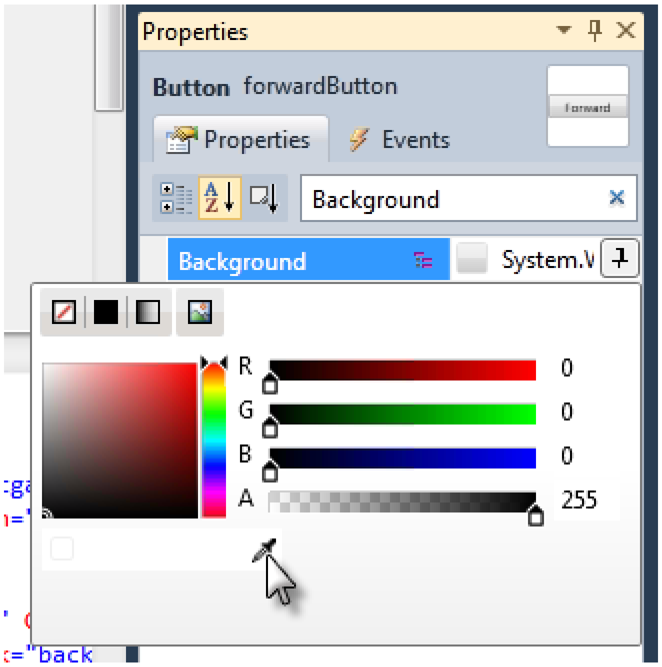


Figure 19

Setting Background Property

4. Click the Background property and the Brush editor will appear. Click the eyedropper tool and select a color from anywhere on your screen. You will see the background of the button has changed

5. Now click the black square next to background label in the property browser to pick the "Extract to Resource" option.

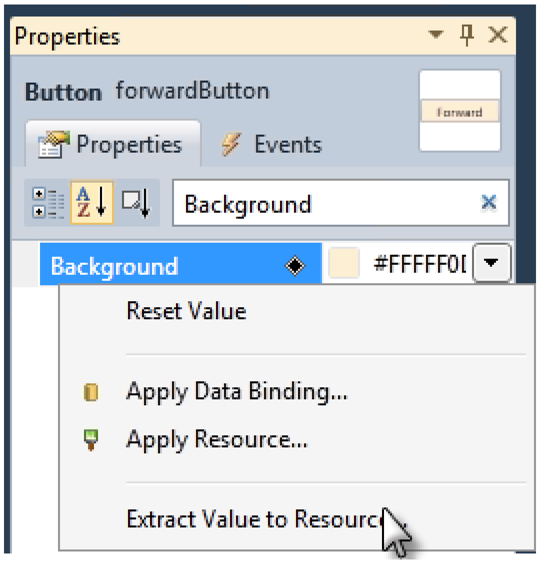


Figure 20

Extract Value to Resource

6. Click **OK** in the Create Resource dialogue, accepting the default values. This window lets you extract the property's value into a resource that you can re-use. In this case, the resource will be created in Window1.xaml.
7. Look in the XAML code editor and you'll find a SolidColorBrush in <Window.Resources />

XAML

```
<Window.Resources>
    <!-- Additional Resources Omitted -->
    <SolidColorBrush x:Key="LightOrangeBrushKey">#FFFFF0D0</SolidColorBrush>
</Window.Resources>
```

Task 2 – Using the Resource to Update Background Property of Another Button

1. Select the **backButton** and locate the Background property in the properties windows.
2. Click the black square in the properties window next to the Background property label and select **Apply Resource...**

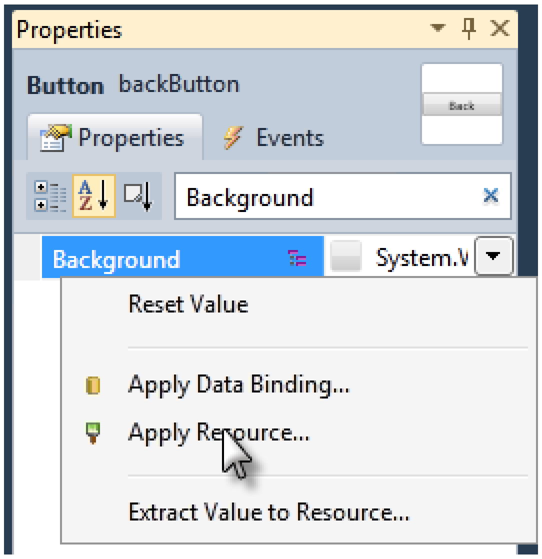


Figure 21
Apply Resource

3. Select the resource from the **Local** expander

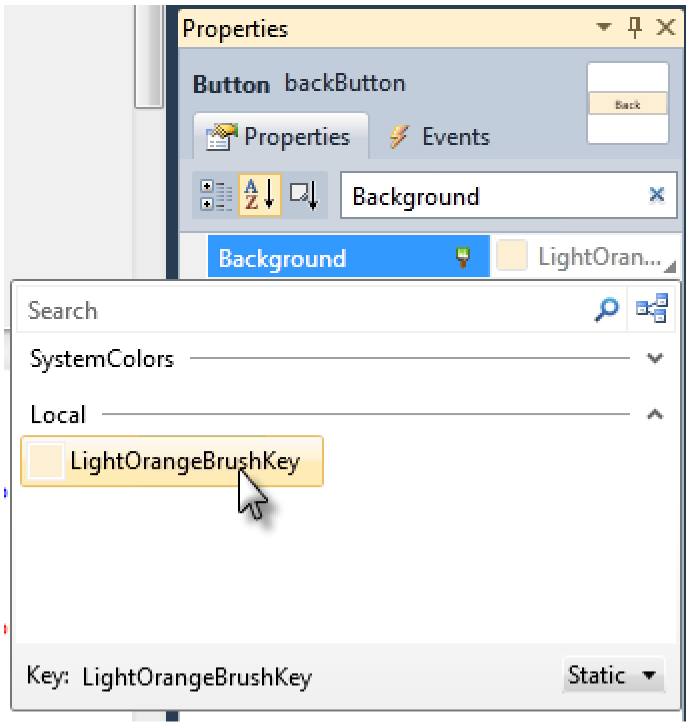


Figure 22
Apply Resource

4. The **forwardButton** and **backButton** now share the same resource, defining the Background.

Next Step

[Summary]

Summary

In this lab you have learned how to create and customize a master-detail business form in WPF 4.0. You have created data scaffolding for the application both simple and master-detail.

Among other features, you have seen how to customize the output of datasources to create custom visuals using the Databinding expression builder from the property browser, and used markup extension intellisense to create databinding expressions.

Moreover, you have seen how to extract a common look into a resource and how to apply it using the resource picker.

All these tools, platform, and languages in the latest release of Visual Studio 2010 should boost your ability to create powerful master-detailed business applications.