Lab Answer Key: Module 9: Designing the User Interface for a Graphical Application

Lab: Customizing Student Photographs and Styling the Application

Exercise 1: Customizing the Appearance of Student Photographs

Task 1: Create the StudentPhoto user control

1. Start the MSL-TMG1 virtual machine if it is not already running.
2. Start the 20483B-SEA-DEV11 virtual machine.
3. Log on to Windows 8 as **Student** with the password **Pa$$w0rd**. If necessary, click **Switch User** to display the list of users.
4. Switch to the Windows 8 Start window and then type Explorer.
5. In the **Apps** list, click **File Explorer**.
6. Navigate to the **E:\Mod09\Labfiles\Databases** folder, and then double-click **SetupSchoolGradesDB.cmd**.
7. Close File Explorer.
8. Switch to the Windows 8 **Start** window.
9. Click **Visual Studio 2012**.
10. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
11. In the **Open Project** dialog box, browse to **E:\Mod09\Labfiles\Starter\Exercise 1**, click **Grades.sln**, and then click **Open**.
12. In Solution Explorer, right-click **Solution 'Grades'**, and then click **Properties**.

13. On the **Startup Project** page, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.

14. In Solution Explorer, expand **Grades.WPF**, and then expand **Controls**.

15. Right-click **Controls**, point to **Add**, and then click **New Item**.

16. In the **Add New Item – Grades.WPF** dialog box, in the template list, click **User Control (WPF)**.

17. In the **Name** box, type **StudentPhoto**, and then click **Add**.

18. In the XAML editor, modify the markup to look like the following markup (the changes are highlighted as bold text):

```
<UserControl x:Class="Grades.WPF.StudentPhoto"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentatio
n"

xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
            xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
             mc:Ignorable="d"
             d:DesignHeight="300" d:DesignWidth="300">
    <Grid>
        <Image Stretch="UniformToFill" Source="{Binding File}"
Margin="8"  />
        <Image Margin="0" Source="../Images/Image_Frame.png"
Stretch="Fill" />
        <TextBlock Text="{Binding Name}" Style="
{StaticResource LabelCenter}"
FontSize="16" VerticalAlignment="Bottom"
Margin="8,0,14.583,8" />     </Grid>
</UserControl>
```

19. In Solution Explorer, expand **StudentPhoto.xaml**, and then double-click **StudentPhoto.xaml.cs**.

20. In the code editor, delete all of the **using** directives, and then type the following code:

```
using System.Windows.Controls;
using System.Windows.Media.Animation;
```

21. Modify the **namespace Grades.WPF.Controls** code to look like the following code:

```
namespace Grades.WPF
```

Task 2: Display the students' photographs in the StudentsPage view

1. In Solution Explorer, expand **Views**, and then double-click **StudentsPage.xaml**.

2. Locate the **<!-- TODO: Exercise 1: Task 2a: Define the DataTemplate for the "list" ItemsControl including the StudentPhoto user control -->** comment, click at the end of the comment, press Enter, and then type the following markup:

```
<ItemsControl.ItemTemplate>
    <DataTemplate>
        <Grid Margin="8">
            <local:StudentPhoto Height="150" Width="127.5"
Cursor="Hand"
```

> **Note:** When you type an opening tag for an element, such as **<ItemsControl.ItemTemplate>**, the XAML editor automatically creates a corresponding closing tag, such as **</ItemsControl.ItemTemplate>**. For the purposes of these instructions, and to ensure that code appears in the correctly commented place, delete any closing tags that are generated automatically; you will add them in at the appropriate point in the XAML markup in later steps.

Task 3: Enable the user to display the details for a student

1. In **StudentsPage.xaml**, locate the **<!-- TODO: Exercise 1: Task 3a: Set the handler for the click event for the StudentPhoto control -->** comment, and above the comment, click at the end of the **Cursor="Hand"** markup.
2. Press Spacebar, and then type the following markup:

```
MouseLeftButtonUp="Student_Click" />
```

3. In the **Task List** window, in the **Categories** list, click **Comments**.
4. Double-click the **TODO: Exercise 1: Task 3b: Review the following event handler.** task.
5. Review the **Student_Click** method which raises the **StudentSelected** event to display the details of the student when a user clicks their photo.

Task 4: Add a Remove button to the StudentsPage view

1. In StudentsPage.xaml, Locate the **<!-- TODO: Exercise 1: Task 4a: Add the "Remove" button to the DataTemplate -->** comment, click at the end of the comment, press Enter, and then type the following markup:

```
<Grid VerticalAlignment="Top" HorizontalAlignment="Right"
Background="#00000000"
Opacity="0.3" Width="20" Height="20"
ToolTipService.ToolTip="Remove from class"
Tag="{Binding}" >
    <Image Source="../Images/delete.png" Stretch="Uniform" />
</Grid>
```

2. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4b: Review the following event handler** task.

3. The code in this method increases the opacity of the grid containing the remove button and reduces the opacity of the grid containing the photo when the user moves the mouse over the delete image

4. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4c: Review the following event handler** task.

5. The code in this method reduces the opacity of the grid containing the remove button and increases the opacity of the grid containing the photo when the user moves the mouse away from the delete image.

6. In the **Task List** window, double-click the **TODO: Exercise 1: Task 4d: Review the following event handler** task.

7. The code in this method removes a student from the current teacher's class when a user clicks the remove icon.

8. In StudentsPage.xaml, locate the **<!-- TODO: Exercise 1: Task 4d: Add event handlers to highlight the "Remove" button as the mouse enters and exits this control -->** comment, and above the comment, click at the end of the **Tag="{Binding}"** code (before the closing **>** tag), press Enter, and then type the following markup:

```
MouseEnter="RemoveStudent_MouseEnter"
MouseLeave="RemoveStudent_MouseLeave"
MouseLeftButtonUp="RemoveStudent_Click"
```

9. Click at the end of the **<Image Source="../Images/delete.png" Stretch="Uniform" /></Grid>** markup, press Enter, and then type the following code:

```
            </Grid>
        </DataTemplate>
    </ItemsControl.ItemTemplate>
```


Task 5: Display all students for the current teacher

1. In the **Task List** window, double-click the **TODO: Exercise 1: Task 5a: Bind the list of students to the "list" ItemsControl** task.
2. In this method, review the code which finds all students for the current teacher and constructs a list of students.
3. In the **Task List** window, double-click the **TODO: Exercise 1: Task 5a: Bind the list of students to the "list" ItemsControl** task.
4. In the code editor, click in the blank line below the comment, and then type the following code:

```
list.ItemsSource = resultData;
```


Task 6: Build and test the application

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging**.
3. When the application starts, in the **Username** box, type **vallee**, in the

Password box, type **password99**, and then click **Log on**.

4. Verify that the students list appears with photographs.
5. In the student list, hover over the **red x** for the student Martin Weber.
6. Verify that the student photograph for Martin Weber becomes transparent and that the red x becomes opaque.
7. Move the cursor away from the red x and verify that the student photograph becomes opaque and that the red x becomes transparent.
8. Click the **red x** for Martin Weber, verify that the **Student** message box appears, and then click **Yes**.
9. Verify that Martin Weber is removed from the student list.
10. Close the application, and then close Visual Studio.

Results: After completing this exercise, the application will display the photographs of each student on the Student List page.

Exercise 2: Styling the Logon View

Task 1: Define and apply styles for the LogonPage view

1. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.
2. In the **Open Project** dialog box, browse to **E:\Mod09\Labfiles\Starter\Exercise 2**, click **Grades.sln**, and then click **Open**.
3. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.
4. On the **Startup Project** page, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.

5. In Solution Explorer, expand **Grades.WPF**, expand **Views**, and then double-click **LogonPage.xaml**.

6. In the XAML editor, locate the **<!-- TODO: Exercise 2: Task 1a: Define the LoginTextBoxStyle -->** comment.

7. Click at the end of the comment, press Enter, and then type the following markup:

```
<UserControl.Resources>
    <Style x:Key="LoginTextBoxStyle" BasedOn="{StaticResource
TextBoxStyle}"
TargetType="{x:Type TextBox}">
        <Setter Property="Margin" Value="5" />
        <Setter Property="FontSize" Value="24"/>
        <Setter Property="MaxLength" Value="16" />
    </Style>
```

8. Locate the **<!-- TODO: Exercise 2: Task 1b: Apply the LoginTextBoxStyle to the "username" TextBox -->** comment.

9. In the line of markup below the comment, delete the **FontSize** property, and then modify the markup as shown in bold below:

```
<TextBox x:Name="username" Grid.Row="1" Grid.Column="1" Style="
{StaticResource
LoginTextBoxStyle}" />
```

10. Locate the **<!-- TODO: Exercise 2: Task 1c: Define the PasswordBoxStyle -->** comment.

11. Click at the end of the comment, press Enter, and then type the following markup:

```
    <Style x:Key="PasswordBoxStyle" TargetType="{x:Type
PasswordBox}">
```

```
            <Setter Property="Margin" Value="5" />
            <Setter Property="FontSize" Value="24"/>
            <Setter Property="MaxLength" Value="16" />
        </Style>
    </UserControl.Resources>
```

12. Locate the **TODO: Exercise 2: Task 1d: Apply the PasswordBoxStyle to the "password" TextBox** comment.
13. In the line of markup below the comment, delete the **FontSize** property, and then modify the markup as shown in bold below:

```
<PasswordBox x:Name="password" Grid.Row="2" Grid.Column="1"
Style="{StaticResource
PasswordBoxStyle}" />
```

Task 2: Define global styles for the application

1. In Solution Explorer, expand **Themes**, and then double-click **Generic.xaml**.
2. In the XAML editor, locate the **<!-- TODO: Exercise 2: Task 2a: Define the label styling used throughout the application -->** comment near the end of the file.
3. Click in the blank line below the comment, and type the following markup:

```
<Setter Property="TextWrapping" Value="NoWrap"/>
<Setter Property="FontFamily" Value="Buxton Sketch"/>
<Setter Property="FontSize" Value="19"/>
<Setter Property="Foreground" Value="#FF303030" />
```

4. Locate the **<!-- TODO: Exercise 2: Task 2b: Define the text styling used**

throughout the application --> comment.

5. Click in the blank line below the comment, then type the following markup:

```
<Setter Property="TextWrapping" Value="NoWrap"/>
<Setter Property="FontFamily" Value="Buxton Sketch"/>
<Setter Property="FontSize" Value="12"/>
<Setter Property="TextAlignment" Value="Left" />
<Setter Property="Foreground" Value="#FF303030" />
```


Task 3: Build and test the application

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging.**
3. When the application starts, in the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
4. In **The School of Fine Arts** window, verify that the styling of the text elements of the application has changed.

Comparison of the Logon views

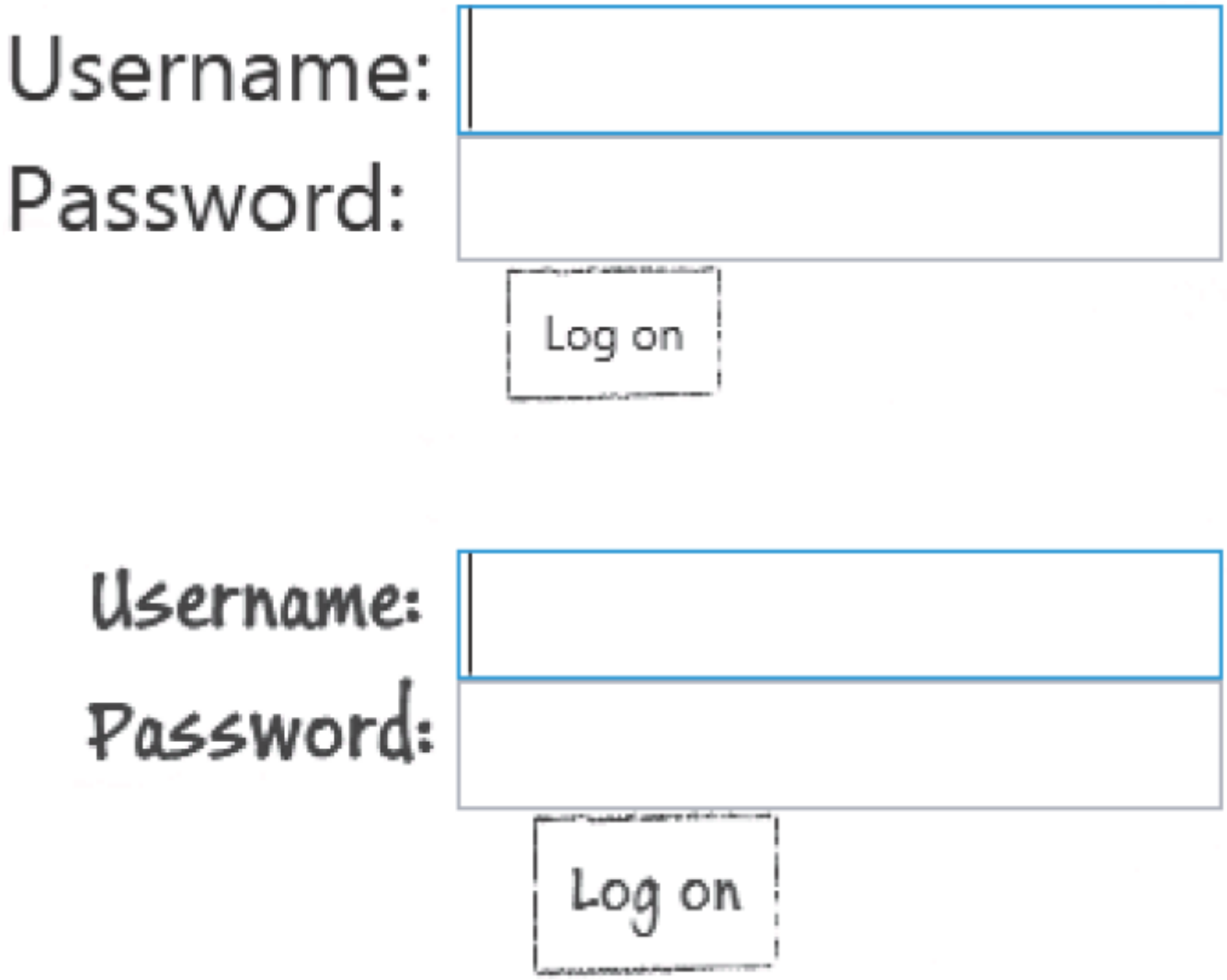

FIGURE 9.1:UPPER: OLD STYLE LOGON VIEW. LOWER: NEW STYLE LOGON VIEW

5. Close the application.

6. On the **File** menu, click **Close Solution**.

Results: After completing this exercise, the Logon view will be styled with a consistent look and feel.

Exercise 3: Animating the StudentPhoto Control (If Time Permits)

Task 1: Define animations for the StudentPhoto control

1. In Visual Studio, on the **File** menu, point to **Open**, and then click **Project/Solution**.

2. In the **Open Project** dialog box, browse to **E:\Mod09\Labfiles\Starter\Exercise 3**, click **Grades.sln**, and then click **Open**.

3. In Solution Explorer, right-click **Solutions 'Grades'**, and then click **Properties**.

4. On the **Startup Project** page, click **Multiple startup projects**. Set **Grades.Web** and **Grades.WPF** to **Start without debugging**, and then click **OK**.

5. In Solution Explorer, in the **Grades.WPF** project, expand **Controls**, and then double-click **StudentPhoto.xaml**.

6. In the XAML editor, locate the **<!-- TODO: Exercise 3: Task 1a: Define a ScaleTransform called "scale" -->** comment.

7. Click in the blank line below the comment, then type the following markup:

```
<UserControl.RenderTransform>
    <ScaleTransform x:Name="scale" />
</UserControl.RenderTransform>
```

8. In the XAML editor, locate the **<!-- TODO: Exercise 3: Task 1b: Define animations for the "scale" transform-->** comment.

9. Click in the blank line below the comment, then type the following markup.

```
<UserControl.Resources>
    <Storyboard x:Key="sbMouseEnter">
        <DoubleAnimation To="1.1" BeginTime="00:00:00"
Duration="00:00:00.05"
Storyboard.TargetName="scale"
Storyboard.TargetProperty="ScaleX" />
        <DoubleAnimation To="1.1" BeginTime="00:00:00"
Duration="00:00:00.15"
Storyboard.TargetName="scale"
```

```
Storyboard.TargetProperty="ScaleY" />
    </Storyboard>
    <Storyboard x:Key="sbMouseLeave">
        <DoubleAnimation To="1" BeginTime="00:00:00"
Duration="00:00:00.05"
Storyboard.TargetName="scale"
Storyboard.TargetProperty="ScaleX" />
        <DoubleAnimation To="1" BeginTime="00:00:00"
Duration="00:00:00.15"
Storyboard.TargetName="scale"
Storyboard.TargetProperty="ScaleY" />
    </Storyboard>
</UserControl.Resources>
```


Task 2: Add event handlers to trigger the animations

1. In Solution Explorer, expand **StudentPhoto.xaml**, and then double-click **StudentPhoto.xaml.cs**.
2. In the **Task List** window, double-click the **TODO: Exercise 3: Task 2a: Handle mouse events to trigger the storyboards that animate the photograph** task.
3. In the code editor, click in the blank line below the comment, and then type the following code:

```
public void OnMouseEnter()
{
    // Trigger the mouse enter animation to grow the size of
the photograph currently
under the mouse pointer
    (this.Resources["sbMouseEnter"] as Storyboard).Begin();
}
public void OnMouseLeave()
{
```

```
    // Trigger the mouse leave animation to shrink the size of
the photograph
currently under the mouse pointer to return it to its original
size


    (this.Resources["sbMouseLeave"] as Storyboard).Begin();
}
```

4. In Solution Explorer, in **Views**, expand **StudentsPage.xaml**, and then double-click **StudentsPage.xaml.cs**.

5. In the **Task List** window, double-click the **TODO: Exercise 3: Task 2b: Forward the MouseEnter and MouseLeave events to the photograph control** task.

6. In the code editor, click in the blank space below the comment, and then type the following code:

```
private void Student_MouseEnter(object sender, MouseEventArgs
e)
{
    // Call the OnMouseEnter event handler on the specific
photograph currently under
the mouse pointer
    ((StudentPhoto)sender).OnMouseEnter();
}
private void Student_MouseLeave(object sender, MouseEventArgs
e)
{
    // Call the OnMouseLeave event handler on the specific
photograph currently under
the mouse pointer
    ((StudentPhoto)sender).OnMouseLeave();
}
```

7. In Solution Explorer, double-click **StudentsPage.xaml**.

8. In the XAML editor, locate the **<!-- TODO: Exercise 3: Task 2c: Specify the handlers for the MouseEnter and MouseLeave events -->** comment.

9. Below the comment, click at the end of the **MouseLeftButtonUp="Student_Click"** markup, press Spacebar, and then type the following markup:

```
MouseEnter="Student_MouseEnter" MouseLeave="Student_MouseLeave"
```


Task 3: Build and test the application

1. On the **Build** menu, click **Build Solution**.
2. On the **Debug** menu, click **Start Without Debugging.**
3. When the application starts, in the **Username** box, type **vallee**, and in the **Password** box, type **password99**, and then click **Log on**.
4. Hover over one of the students in the student list and verify that the photograph animates—it should expand and contract as the mouse passes over it.
5. Close the application.
6. On the **File** menu, click **Close Solution**.

Results: After completing this exercise, the Photograph control will be animated.