Hands-On Lab

Settings and Preferences

Lab version: 1.0.0

Last updated: 6/9/2012



Contents

[Overview 3](#_Toc319161363)

[Exercise 1: Add an About Page 4](#_Toc319161364)

[Task 1 – Add a SettingsFlyout Class 4](#_Toc319161365)

[Task 2 – Add an About Command 5](#_Toc319161366)

[Task 3 – Add an About Page 7](#_Toc319161367)

[Task 4 – Test the Results 12](#_Toc319161368)

[Exercise 2: Add a Preferences Page 13](#_Toc319161369)

[Task 1 – Add a Preferences Command 13](#_Toc319161370)

[Task 2 – Add a Preferences Page 14](#_Toc319161371)

[Task 3 – Make the Preference Sticky 19](#_Toc319161372)

[Exercise 3: Implement the Preference 21](#_Toc319161373)

[Task 1 – Modify the OnLaunched Method 21](#_Toc319161374)

[Task 2 – Test the Results 21](#_Toc319161375)

[Summary 22](#_Toc319161376)

Overview

* 1. Lab 3 introduced Metro’s charms bar and demonstrated how easily applications can integrate with the Share and Search charms. The charms bar also includes a Settings charm, which is used to change settings in the active Metro application. In the settings pane that appears when you select the Settings charm, the operating system provides a Permissions command that allows users to enable and disable certain features of the program such as webcam and microphone access. Significantly, you can add commands of your own to the settings pane and connect them to settings pages, providing users with convenient access to preferences, about boxes, and other application-specific settings content.
  2. In this lab, you’ll add About and Preferences commands to the settings pane in Contoso Cookbook. You’ll expose a simple user preference that can be toggled on and off with a checkbox, and you’ll use roaming settings to store that preference so it will follow users wherever they go.

# Objectives

* 1. This lab will show you how to:
  + Add an About command and an about page to the settings pane
  + Add a Preferences command and a preferences page to the settings pane
  + Use roaming settings to store user preferences

# System Requirements

* 1. You must have the following items to complete this lab:
  + Microsoft Windows 8 Release Preview
  + Microsoft Visual Studio 2012 RC for Windows 8

# Setup

* 1. You must perform the following steps to prepare your computer for this lab:
  2. Install the Microsoft Windows 8 Release Preview
  3. Install the Microsoft Visual Studio 2012 RC for Windows 8

# Exercises

* 1. This Hands-On Lab comprises the following exercises:
  2. Add an About Page
  3. Add a Preferences Page
  4. Implement the Preference
  5. Estimated time to complete this lab: **30 to 40 minutes**.

Exercise 1: Add an About Page

1. In this exercise, you’ll add a simple about page to Contoso Cookbook.

Task 1 – Add a SettingsFlyout Class

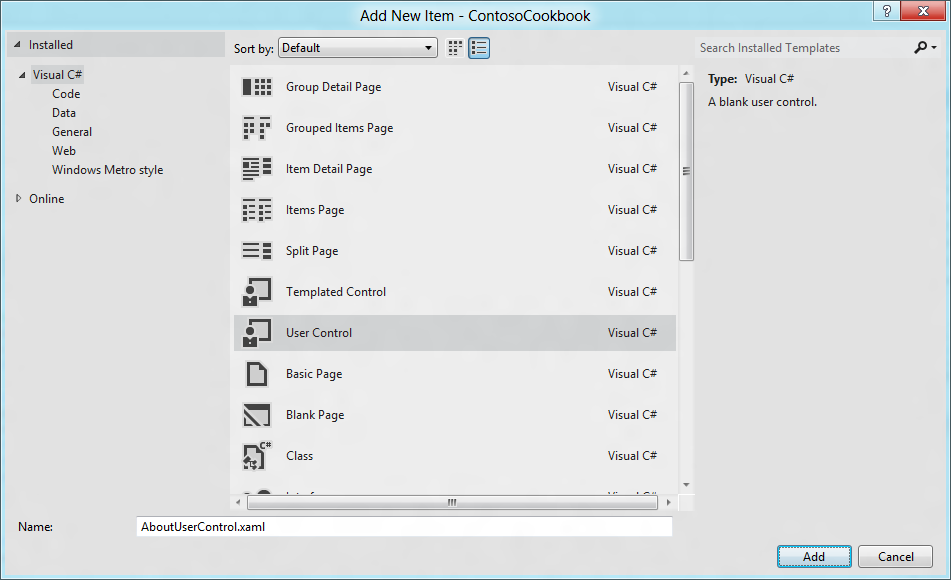
* 1. The first step is to add a class representing settings flyouts (the UI elements that sweep in from the right side of the screen when you select a command in the settings pane). WinRT doesn’t provide that class for you, but it’s easy enough to write one of your own.
  2. Open the ContosoCookbook project you finished in Lab 5 in Visual Studio. If you didn’t complete Lab 5 or would like to start with a reference copy, you’ll find a completed version of the lab in the starting materials.
  3. Right-click the project’s Common folder and use the **Add - New Item** command to add a new class to the project. Name the file SettingsFlyout.cs.
  4. Replace the file’s contents with this:
     1. C#
     2. using System;
     3. using System.Collections.Generic;
     4. using System.Linq;
     5. using System.Text;
     6. using System.Threading.Tasks;
     7. using Windows.UI.Xaml;
     8. using Windows.UI.Xaml.Controls;
     9. using Windows.UI.Xaml.Controls.Primitives;
     10. namespace ContosoCookbook.Common
     11. {
     12. class SettingsFlyout
     13. {
     14. private const int \_width = 346;
     15. private Popup \_popup;
     16. public void ShowFlyout(UserControl control)
     17. {
     18. \_popup = new Popup();
     19. \_popup.Closed += OnPopupClosed;
     20. Window.Current.Activated += OnWindowActivated;
     21. \_popup.IsLightDismissEnabled = true;
     22. \_popup.Width = \_width;
     23. \_popup.Height = Window.Current.Bounds.Height;
     24. control.Width = \_width;
     25. control.Height = Window.Current.Bounds.Height;
     26. \_popup.Child = control;
     27. \_popup.SetValue(Canvas.LeftProperty, Window.Current.Bounds.Width - \_width);
     28. \_popup.SetValue(Canvas.TopProperty, 0);
     29. \_popup.IsOpen = true;
     30. }
     31. private void OnWindowActivated(object sender, Windows.UI.Core.WindowActivatedEventArgs e)
     32. {
     33. if (e.WindowActivationState == Windows.UI.Core.CoreWindowActivationState.Deactivated)
     34. {
     35. \_popup.IsOpen = false;
     36. }
     37. }
     38. void OnPopupClosed(object sender, object e)
     39. {
     40. Window.Current.Activated -= OnWindowActivated;
     41. }
     42. }
     43. }
  5. Save your changes and close SettingsFlyout.cs.

Task 2 – Add an About Command

* 1. The next step is to add an About command to the settings menu, which we accomplish by handling CommandsRequested events.
  2. Open App.xaml.cs and add the following using statements:
     1. C#
     2. using Windows.UI.ApplicationSettings;
     3. using ContosoCookbook.Common;
  3. Now add the following statements to the end of the OnLaunched method:
     1. C#
     2. // Register handler for CommandsRequested events from the settings pane
     3. SettingsPane.GetForCurrentView().CommandsRequested += OnCommandsRequested;
  4. SearchResultsPage.xaml를 열어서 가장 상단에 using 문을 넣어준다.
     1. C#
     2. using Windows.UI.ApplicationSettings;
  5. SearchResultsPage.xaml를 열어서Activate 메서드를 찾아서 아래 코드를 넣는다. 검색을 통해서 앱이 Activate 될 때도 이벤트 핸들러를 등록해주는 것이다. :
     1. C#
     2. public static void Activate(String queryText, ApplicationExecutionState previousExecutionState)
     3. {
     4. var previousContent = Window.Current.Content;
     5. var frame = previousContent as Frame;
     6. if (frame != null)
     7. {
     8. // If the app is already running and uses top-level frame navigation we can just
     9. // navigate to the search results
     10. frame.Navigate(typeof(SearchResultsPage), queryText);
     11. }
     12. else
     13. {
     14. frame = new Frame();
     15. Window.Current.Content = frame;
     16. SettingsPane.GetForCurrentView().CommandsRequested += (Application.Current as App).OnCommandsRequested;
     17. //frame.Navigate(typeof(SearchResultsPage), new Tuple<String, UIElement>(queryText, previousContent));
     18. frame.Navigate(typeof(SearchResultsPage), queryText);
     19. }
     20. // Either way, active the window
     21. Window.Current.Activate();
     22. }
  6. Add the following event handler to App.xaml.cs:
     1. C#
     2. public void OnCommandsRequested(SettingsPane sender, SettingsPaneCommandsRequestedEventArgs args)
     3. {
     4. // Add an About command
     5. var about = new SettingsCommand("about", "About", (handler) =>
     6. {
     7. var settings = new SettingsFlyout();
     8. settings.ShowFlyout(new AboutUserControl());
     9. });
     11. args.Request.ApplicationCommands.Add(about);
     12. }
     13. **Note:** Adding a command to the settings menu is accomplished by adding a SettingsCommand object to the ApplicationCommands collection passed to CommandsRequested events. The third parameter to the SettingsCommand constructor is the handler that’s called when the command is invoked. In this example, you’re using the SettingsFlyout class you added in the previous task to display an about page from the handler. Of course, that page doesn’t exist yet. You’ll take care of that in the next task.

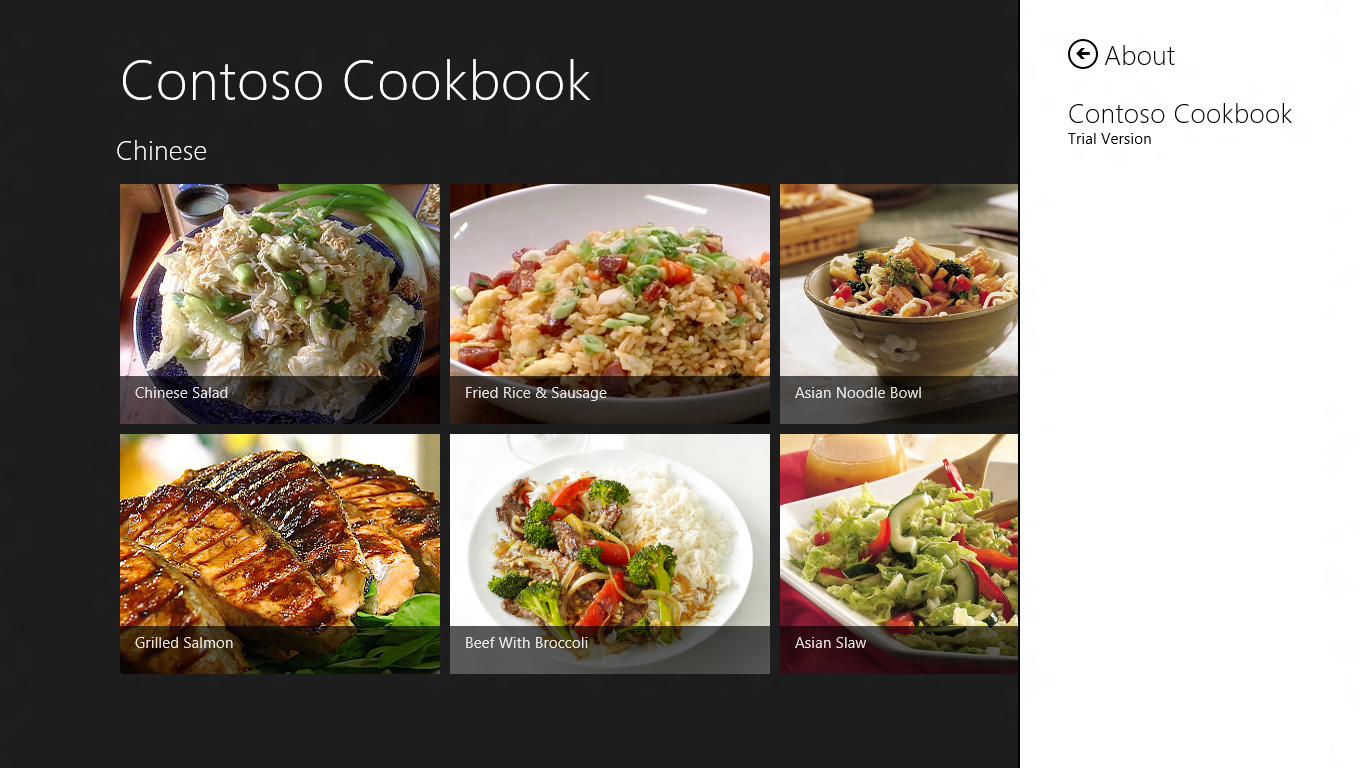
Task 3 – Add an About Page

The event handler you added in the previous task adds an About command to the settings pane. The next step is to add an about page for the About command to display. To create that page, we’ll add a new user control to the project.

* 1. In Solution Explorer, right-click the project and use the **Add - New Item** command to add a user control to the project. Name the file AboutUserControl.xaml, as shown in Figure 1.
  2. 
  3. Figure 1
  4. Adding a user control representing the about page
  5. Replace the contents of AboutUserControl.xaml with the following statements:
     1. XAML
     2. <UserControl
     3. x:Class="ContosoCookbook.AboutUserControl"
     4. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
     5. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
     6. xmlns:local="using:ContosoCookbook"
     7. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
     8. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
     9. mc:Ignorable="d"
     10. d:DesignHeight="768"
     11. d:DesignWidth="346">
     13. <Border BorderBrush="Black" BorderThickness="1,0,0,0">
     14. <Grid Background="White" VerticalAlignment="Stretch">
     15. <!-- Root grid definition -->
     16. <Grid.RowDefinitions>
     17. <RowDefinition Height="80" />
     18. <RowDefinition Height="\*" />
     19. </Grid.RowDefinitions>
     20. <!-- Header area for panel -->
     21. <Grid Grid.Row="0">
     22. <Grid Margin="30,32,17,13">
     23. <Grid.Transitions>
     24. <TransitionCollection>
     25. <EntranceThemeTransition FromHorizontalOffset="50" />
     26. </TransitionCollection>
     27. </Grid.Transitions>
     28. <Grid.ColumnDefinitions>
     29. <ColumnDefinition Width="30" />
     30. <ColumnDefinition Width="\*" />
     31. </Grid.ColumnDefinitions>
     32. <Button Click="OnBackButtonClicked" Margin="0,3,0,0" Grid.Column="0" Style="{StaticResource SettingsBackButtonStyle}" HorizontalAlignment="Left" />
     33. <TextBlock Foreground="Black" Margin="10,0,0,0" Grid.Column="1" FontFamily="Segoe UI" FontWeight="SemiLight" FontSize="26.6667" Text="About" HorizontalAlignment="Left" />
     34. </Grid>
     35. </Grid>
     36. <!-- Settings Panel Content -->
     37. <Grid Grid.Row="1" Margin="28,12,23,0" VerticalAlignment="Top">
     38. <Grid.Transitions>
     39. <TransitionCollection>
     40. <EntranceThemeTransition FromHorizontalOffset="120" />
     41. </TransitionCollection>
     42. </Grid.Transitions>
     43. <StackPanel Orientation="Vertical">
     44. <TextBlock Text="Contoso Cookbook" Foreground="Black" FontFamily="Segoe UI" FontWeight="SemiLight" FontSize="26.667" />
     45. <TextBlock Text="Trial Version" Foreground="Black" FontFamily="Segoe UI" FontWeight="SemiLight" FontSize="18" />
     46. </StackPanel>
     47. </Grid>
     48. </Grid>
     49. </Border>
     50. </UserControl>
  6. Open AboutUserControl.xaml.cs and add the following using statement:
     1. C#
     2. using Windows.UI.ApplicationSettings;
  7. Then add the following method:
     1. C#
     2. private void OnBackButtonClicked(object sender, RoutedEventArgs e)
     3. {
     4. if (this.Parent.GetType() == typeof(Popup))
     5. {
     6. ((Popup)this.Parent).IsOpen = false;
     7. }
     8. SettingsPane.Show();
     9. }
     10. **Note:** The about page currently informs the user that this is a trial version of Contoso Cookbook. In Lab 8, you’ll use the Windows Runtime’s store APIs to simulate purchases of the app, and once a purchase occurs, you’ll replace “Trial Version” with licensing information.
  8. Finish up by opening StandardStyles.xaml and adding this style resource for the about page’s back button:
     1. XAML
     2. <Style x:Key="SettingsBackButtonStyle" TargetType="Button">
     3. <Setter Property="MinWidth" Value="0"/>
     4. <Setter Property="FontFamily" Value="Segoe UI Symbol"/>
     5. <Setter Property="FontWeight" Value="Normal"/>
     6. <Setter Property="FontSize" Value="26.66667"/>
     7. <Setter Property="AutomationProperties.AutomationId" Value="BackButton"/>
     8. <Setter Property="AutomationProperties.Name" Value="Back"/>
     9. <Setter Property="AutomationProperties.ItemType" Value="Navigation Button"/>
     10. <Setter Property="Template">
     11. <Setter.Value>
     12. <ControlTemplate TargetType="Button">
     13. <Grid x:Name="RootGrid" Width="30" Height="30">
     14. <Grid Margin="-6,-6,0,0">
     15. <TextBlock x:Name="BackgroundGlyph" Text="&#xE0D4;" Foreground="{StaticResource BackButtonBackgroundThemeBrush}"/>
     16. <TextBlock x:Name="NormalGlyph" Text="{StaticResource BackButtonGlyph}" Foreground="{StaticResource BackButtonForegroundThemeBrush}"/>
     17. <TextBlock x:Name="ArrowGlyph" Text="&#xE0C4;" Foreground="{StaticResource BackButtonPressedForegroundThemeBrush}" Opacity="0"/>
     18. </Grid>
     19. <Rectangle
     20. x:Name="FocusVisualWhite"
     21. IsHitTestVisible="False"
     22. Stroke="{StaticResource FocusVisualWhiteStrokeThemeBrush}"
     23. StrokeEndLineCap="Square"
     24. StrokeDashArray="1,1"
     25. Opacity="0"
     26. StrokeDashOffset="1.5"/>
     27. <Rectangle
     28. x:Name="FocusVisualBlack"
     29. IsHitTestVisible="False"
     30. Stroke="{StaticResource FocusVisualBlackStrokeThemeBrush}"
     31. StrokeEndLineCap="Square"
     32. StrokeDashArray="1,1"
     33. Opacity="0"
     34. StrokeDashOffset="0.5"/>
     35. <VisualStateManager.VisualStateGroups>
     36. <VisualStateGroup x:Name="CommonStates">
     37. <VisualState x:Name="Normal" />
     38. <VisualState x:Name="PointerOver">
     39. <Storyboard>
     40. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="BackgroundGlyph" Storyboard.TargetProperty="Foreground">
     41. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource BackButtonPressedForegroundThemeBrush}"/>
     42. </ObjectAnimationUsingKeyFrames>
     43. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="NormalGlyph" Storyboard.TargetProperty="Foreground">
     44. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource BackButtonPointerOverForegroundThemeBrush}"/>
     45. </ObjectAnimationUsingKeyFrames>
     46. </Storyboard>
     47. </VisualState>
     48. <VisualState x:Name="Pressed">
     49. <Storyboard>
     50. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="BackgroundGlyph" Storyboard.TargetProperty="Foreground">
     51. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource BackButtonForegroundThemeBrush}"/>
     52. </ObjectAnimationUsingKeyFrames>
     53. <DoubleAnimation
     54. Storyboard.TargetName="ArrowGlyph"
     55. Storyboard.TargetProperty="Opacity"
     56. To="1"
     57. Duration="0"/>
     58. <DoubleAnimation
     59. Storyboard.TargetName="NormalGlyph"
     60. Storyboard.TargetProperty="Opacity"
     61. To="0"
     62. Duration="0"/>
     63. </Storyboard>
     64. </VisualState>
     65. <VisualState x:Name="Disabled">
     66. <Storyboard>
     67. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="RootGrid" Storyboard.TargetProperty="Visibility">
     68. <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
     69. </ObjectAnimationUsingKeyFrames>
     70. </Storyboard>
     71. </VisualState>
     72. </VisualStateGroup>
     73. <VisualStateGroup x:Name="FocusStates">
     74. <VisualState x:Name="Focused">
     75. <Storyboard>
     76. <DoubleAnimation
     77. Storyboard.TargetName="FocusVisualWhite"
     78. Storyboard.TargetProperty="Opacity"
     79. To="1"
     80. Duration="0"/>
     81. <DoubleAnimation
     82. Storyboard.TargetName="FocusVisualBlack"
     83. Storyboard.TargetProperty="Opacity"
     84. To="1"
     85. Duration="0"/>
     86. </Storyboard>
     87. </VisualState>
     88. <VisualState x:Name="Unfocused" />
     89. <VisualState x:Name="PointerFocused" />
     90. </VisualStateGroup>
     91. </VisualStateManager.VisualStateGroups>
     92. </Grid>
     93. </ControlTemplate>
     94. </Setter.Value>
     95. </Setter>
     96. </Style>

Task 4 – Test the Results

Now it’s time to test your changes and see what a Metro-style about page looks like.

* 1. Press F5 to run the application.
  2. Display the charms bar and select Settings.
  3. Select the About command from the settings pane.
  4. Confirm that the about page appears, as shown in Figure 2.
  5. 
  6. Figure 2
  7. Contoso Cookbook’s about page
  8. Return to Visual Studio and stop debugging.

Exercise 2: Add a Preferences Page

1. Now that you understand the mechanics of adding a page to Metro’s settings pane, you’ll add another page – this time, a preferences page that allows the user to enter and edit preferences. We’ll just add one preference to demonstrate how it’s done, but of course you’re free to add as many preferences as you’d like. The preference you’ll add is one that allows the user to configure Contoso Cookbook to return to the last recipe or recipe group that was displayed when it starts up.

Task 1 – Add a Preferences Command

Start by modifying the CommandsRequested event handler you wrote in the previous exercise –the one that adds an About command to the settings menu – so that it adds a Preferences command, too.

* 1. Open App.xaml.cs and find the OnCommandsRequested method.
  2. Add the highlighted statements to the method:
     1. C#
     2. void OnCommandsRequested(SettingsPane sender, SettingsPaneCommandsRequestedEventArgs args)
     3. {
     4. // Add an About command
     5. var about = new SettingsCommand("about", "About", (handler) =>
     6. {
     7. var settings = new SettingsFlyout();
     8. settings.ShowFlyout(new AboutUserControl());
     9. });
     11. args.Request.ApplicationCommands.Add(about);
     12. // Add a Preferences command
     13. var preferences = new SettingsCommand("preferences", "Preferences", (handler) =>
     14. {
     15. var settings = new SettingsFlyout();
     16. settings.ShowFlyout(new PreferencesUserControl());
     17. });
     18. args.Request.ApplicationCommands.Add(preferences);
     19. }

Task 2 – Add a Preferences Page

The next task is to create the page that the Preferences command invokes.

* 1. In Solution Explorer, right-click the project and use the **Add - New Item** command to add a new user control to the project. Name the file PreferencesUserControl.xaml.
  2. Replace the contents of PreferencesUserControl.xaml with the following statements:
     1. XAML
     2. <UserControl
     3. x:Class="ContosoCookbook.PreferencesUserControl"
     4. xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
     5. xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
     6. xmlns:local="using:ContosoCookbook"
     7. xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
     8. xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
     9. mc:Ignorable="d"
     10. d:DesignHeight="768"
     11. d:DesignWidth="346">
     12. <UserControl.Resources>
     13. <Style x:Key="PreferencesCheckBoxStyle" TargetType="CheckBox">
     14. <Setter Property="Background" Value="Transparent"/>
     15. <Setter Property="Foreground" Value="Black"/>
     16. <Setter Property="MinHeight" Value="50"/>
     17. <Setter Property="Padding" Value="4,0,0,0"/>
     18. <Setter Property="HorizontalAlignment" Value="Stretch"/>
     19. <Setter Property="VerticalAlignment" Value="Center"/>
     20. <Setter Property="HorizontalContentAlignment" Value="Left"/>
     21. <Setter Property="FontFamily" Value="Segoe UI Symbol"/>
     22. <Setter Property="FontSize" Value="22"/>
     23. <Setter Property="Template">
     24. <Setter.Value>
     25. <ControlTemplate TargetType="CheckBox">
     26. <Border BorderBrush="{TemplateBinding BorderBrush}" BorderThickness="{TemplateBinding BorderThickness}" Background="{TemplateBinding Background}">
     27. <VisualStateManager.VisualStateGroups>
     28. <VisualStateGroup x:Name="CommonStates">
     29. <VisualState x:Name="Normal"/>
     30. <VisualState x:Name="PointerOver">
     31. <Storyboard>
     32. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Fill" Storyboard.TargetName="NormalRectangle">
     33. <DiscreteObjectKeyFrame KeyTime="0" Value="White"/>
     34. </ObjectAnimationUsingKeyFrames>
     35. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Stroke" Storyboard.TargetName="NormalRectangle">
     36. <DiscreteObjectKeyFrame KeyTime="0" Value="Black"/>
     37. </ObjectAnimationUsingKeyFrames>
     38. </Storyboard>
     39. </VisualState>
     40. <VisualState x:Name="Pressed">
     41. <Storyboard>
     42. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Fill" Storyboard.TargetName="NormalRectangle">
     43. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource CheckBoxPressedForegroundThemeBrush}"/>
     44. </ObjectAnimationUsingKeyFrames>
     45. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Stroke" Storyboard.TargetName="NormalRectangle">
     46. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource CheckBoxPressedForegroundThemeBrush}"/>
     47. </ObjectAnimationUsingKeyFrames>
     48. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Fill" Storyboard.TargetName="CheckGlyph">
     49. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource CheckBoxPressedBackgroundThemeBrush }"/>
     50. </ObjectAnimationUsingKeyFrames>
     51. </Storyboard>
     52. </VisualState>
     53. <VisualState x:Name="Disabled">
     54. <Storyboard>
     55. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Fill" Storyboard.TargetName="NormalRectangle">
     56. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource CheckBoxDisabledBackgroundThemeBrush }"/>
     57. </ObjectAnimationUsingKeyFrames>
     58. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Stroke" Storyboard.TargetName="NormalRectangle">
     59. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource CheckBoxDisabledBackgroundThemeBrush }"/>
     60. </ObjectAnimationUsingKeyFrames>
     61. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Fill" Storyboard.TargetName="CheckGlyph">
     62. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource CheckBoxDisabledBorderThemeBrush }"/>
     63. </ObjectAnimationUsingKeyFrames>
     64. <ObjectAnimationUsingKeyFrames Storyboard.TargetProperty="Foreground" Storyboard.TargetName="ContentPresenter">
     65. <DiscreteObjectKeyFrame KeyTime="0" Value="{StaticResource CheckBoxContentDisabledForegroundThemeBrush }"/>
     66. </ObjectAnimationUsingKeyFrames>
     67. </Storyboard>
     68. </VisualState>
     69. </VisualStateGroup>
     70. <VisualStateGroup x:Name="CheckStates">
     71. <VisualState x:Name="Checked">
     72. <Storyboard>
     73. <DoubleAnimation Duration="0" To="1" Storyboard.TargetProperty="Opacity" Storyboard.TargetName="CheckGlyph"/>
     74. </Storyboard>
     75. </VisualState>
     76. <VisualState x:Name="Unchecked"/>
     77. <VisualState x:Name="Indeterminate">
     78. <Storyboard>
     79. <DoubleAnimation Duration="0" To="1" Storyboard.TargetProperty="Opacity" Storyboard.TargetName="IndeterminateGlyph"/>
     80. </Storyboard>
     81. </VisualState>
     82. </VisualStateGroup>
     83. <VisualStateGroup x:Name="FocusStates">
     84. <VisualState x:Name="Focused">
     85. <Storyboard>
     86. <DoubleAnimation Duration="0" To="1" Storyboard.TargetProperty="Opacity" Storyboard.TargetName="FocusVisualWhite"/>
     87. <DoubleAnimation Duration="0" To="1" Storyboard.TargetProperty="Opacity" Storyboard.TargetName="FocusVisualBlack"/>
     88. </Storyboard>
     89. </VisualState>
     90. <VisualState x:Name="Unfocused"/>
     91. <VisualState x:Name="PointerFocused"/>
     92. </VisualStateGroup>
     93. </VisualStateManager.VisualStateGroups>
     94. <Grid>
     95. <Grid.ColumnDefinitions>
     96. <ColumnDefinition Width="25"/>
     97. <ColumnDefinition Width="\*"/>
     98. </Grid.ColumnDefinitions>
     99. <Rectangle x:Name="NormalRectangle" Fill="{StaticResource CheckBoxBorderThemeBrush }" Height="21" Stroke="Black" StrokeThickness="1" UseLayoutRounding="True" Width="21"/>
     100. <Path x:Name="CheckGlyph" Data="F1 M 0,58 L 2,56 L 6,60 L 13,51 L 15,53 L 6,64 z" Fill="{StaticResource CheckBoxForegroundThemeBrush}" FlowDirection="LeftToRight" Height="14" Opacity="0" Stretch="Fill" Width="16"/>
     101. <Rectangle x:Name="IndeterminateGlyph" Fill="{StaticResource CheckBoxForegroundThemeBrush}" Height="9" Opacity="0" Width="9"/>
     102. <Rectangle x:Name="FocusVisualWhite" Height="25" Opacity="0" StrokeDashOffset="0.5" StrokeEndLineCap="Square" Stroke="White" StrokeDashArray="1,1" Width="25"/>
     103. <Rectangle x:Name="FocusVisualBlack" Height="25" Opacity="0" StrokeDashOffset="1.5" StrokeEndLineCap="Square" Stroke="Black" StrokeDashArray="1,1" Width="25"/>
     104. <ContentPresenter x:Name="ContentPresenter" ContentTemplate="{TemplateBinding ContentTemplate}" ContentTransitions="{TemplateBinding ContentTransitions}" Content="{TemplateBinding Content}" Grid.Column="1" HorizontalAlignment="{TemplateBinding HorizontalContentAlignment}" Margin="{TemplateBinding Padding}" VerticalAlignment="{TemplateBinding VerticalContentAlignment}"/>
     105. </Grid>
     106. </Border>
     107. </ControlTemplate>
     108. </Setter.Value>
     109. </Setter>
     110. </Style>
     111. </UserControl.Resources>
     112. <Border BorderBrush="Black" BorderThickness="1,0,0,0">
     113. <Grid Background="White" VerticalAlignment="Stretch">
     114. <!-- Root grid definition -->
     115. <Grid.RowDefinitions>
     116. <RowDefinition Height="80" />
     117. <RowDefinition Height="\*" />
     118. </Grid.RowDefinitions>
     119. <!-- Header area for panel -->
     120. <Grid Grid.Row="0">
     121. <Grid Margin="30,32,17,13">
     122. <Grid.Transitions>
     123. <TransitionCollection>
     124. <EntranceThemeTransition FromHorizontalOffset="50" />
     125. </TransitionCollection>
     126. </Grid.Transitions>
     127. <Grid.ColumnDefinitions>
     128. <ColumnDefinition Width="30" />
     129. <ColumnDefinition Width="\*" />
     130. </Grid.ColumnDefinitions>
     131. <Button Click="OnBackButtonClicked" Margin="0,3,0,0" Grid.Column="0" Style="{StaticResource SettingsBackButtonStyle}" HorizontalAlignment="Left" />
     132. <TextBlock Foreground="Black" Margin="10,0,0,0" Grid.Column="1" FontFamily="Segoe UI" FontWeight="SemiLight" FontSize="26.6667" Text="Preferences" HorizontalAlignment="Left" />
     133. </Grid>
     134. </Grid>
     135. <!-- Settings Panel Content -->
     136. <Grid Grid.Row="1" Margin="28,12,23,0" VerticalAlignment="Top">
     137. <Grid.Transitions>
     138. <TransitionCollection>
     139. <EntranceThemeTransition FromHorizontalOffset="120" />
     140. </TransitionCollection>
     141. </Grid.Transitions>
     142. <CheckBox x:Name="Remember" Style="{StaticResource PreferencesCheckBoxStyle}" Content="Remember where I was" Foreground="Black" Click="OnCheckBoxClicked" />
     143. </Grid>
     144. </Grid>
     145. </Border>
     146. </UserControl>
     147. **Note:** Admittedly, that’s a lot of XAML. The bulk of it is a new template for the CheckBox control that sets the color of the rectangle to black so it can be seen against a white background. All that XAML for one tiny change!
  3. Open PreferencesUserControl.xaml.cs and add the following using statement:
     1. C#
     2. using Windows.UI.ApplicationSettings;
  4. Then add the following method:
     1. C#
     2. private void OnBackButtonClicked(object sender, RoutedEventArgs e)
     3. {
     4. if (this.Parent.GetType() == typeof(Popup))
     5. {
     6. ((Popup)this.Parent).IsOpen = false;
     7. }
     8. SettingsPane.Show();
     9. }
  5. Press F5 to run the application.
  6. Display the charms bar and select Settings.
  7. Select the Preferences command from the settings pane.
  8. Confirm that the preferences page appears and that it contains a checkbox, as shown in Figure 3.
  9. 
  10. Figure 3
  11. Contoso Cookbook’s preferences page
  12. Return to Visual Studio and stop debugging.

Task 3 – Make the Preference Sticky

Right now, the “Remember where I was” checkbox in the preferences page isn’t wired up to anything, and it doesn’t retain its state. Let’s fix that by using roaming settings to save the state of the check box, and to initialize the check box each time the preferences page is displayed.

* 1. Open PreferencesUserControl.xaml and add the highlighted Click attribute to the CheckBox control:
     1. XAML
     2. <CheckBox x:Name="Remember" Style="{StaticResource PreferencesCheckBoxStyle}" Content="Remember where I was" Foreground="Black" Click="OnCheckBoxClicked" />
  2. Open PreferencesUserControl.xaml.cs and add the following using statement:
     1. C#
     2. using Windows.Storage;
  3. Then add the following method:
     1. C#
     2. private void OnCheckBoxClicked(object sender, RoutedEventArgs e)
     3. {
     4. // Record the state of the CheckBox in roaming settings
     5. ApplicationData.Current.RoamingSettings.Values["Remember"] = ((CheckBox)sender).IsChecked;
     6. }
  4. Add the following statements to the PreferencesUserControl constructor, after the call to InitializeComponent, to initialize the checkbox each time the preferences page is displayed based on the “Remember” value stored in roaming settings:
     1. C#
     2. // Initialize the CheckBox from roaming settings
     3. if (ApplicationData.Current.RoamingSettings.Values.ContainsKey("Remember"))

Remember.IsChecked = (bool)ApplicationData.Current.RoamingSettings.Values["Remember"];

* 1. Press F5 to run the application.
  2. Display the charms bar and select Settings.
  3. Select the Preferences command from the settings pane.
  4. Tap the “Remember where I was” checkbox to check it.
  5. Dismiss the settings pane.
  6. Return to Visual Studio and stop debugging.
  7. Press F5 to start the application again.
  8. Go to the preferences page and confirm that the checkbox is checked.
  9. Return to Visual Studio and stop debugging.

Exercise 3: Implement the Preference

1. Currently, Contoso Cookbook shows the start page each time it starts up. The purpose of adding a user preference entitled “Remember where I was” in the previous exercise was to allow a user to configure the application to return each time it starts up to the page that was displayed the last time it shut down. Honoring this user preference will require only minor changes to your code since the PLM logic you added in Lab 5 provides most of the infrastructure needed.

Task 1 – Modify the OnLaunched Method

In the previous lab, you wrote PLM code that restores the last location viewed if the app is suspended and then terminated by the operating system. We’ll use a similar strategy to restore the location any time the app starts up if “Remember where I was” is checked.

* 1. Open App.xaml.cs and find the OnLaunched method.
  2. App.xaml.cs 상단에 using 문을 추가한다
     1. C#
     2. using Windows.Storage;
  3. Add the following statements immediately before the if statement that compares args.PreviousExecutionState to ApplicationExecutionState.Terminated:
     1. C#
     2. // Find out if the "Remember where I was" preference is enabled
     3. bool remember = false;
     4. if (ApplicationData.Current.RoamingSettings.Values.ContainsKey("Remember"))
     5. remember = (bool)ApplicationData.Current.RoamingSettings.Values["Remember"];
  4. Modify the if statement that follows by adding the highlighted text:
     1. C#
     2. if (args.PreviousExecutionState == ApplicationExecutionState.Terminated || remember)
     3. {
     4. // Restore the saved session state only when appropriate
     5. await SuspensionManager.RestoreAsync();
     6. }
     7. **Note:** Formerly, you only restored the user’s previous location if the app was suspended and terminated. Now you’re restoring it if the app was suspended and terminated *or* the check box is checked.

Task 2 – Test the Results

All that remains is to do a little testing to ensure that the change works.

* 1. Press F5 to run the application.
  2. Display the charms bar and select Settings.
  3. Select the Preferences command from the settings pane.
  4. Verify that “Remember where I was” is still checked. If it’s not, check it.
  5. Dismiss the settings pane.
  6. Navigate to a recipe page.
  7. Close the application by swiping downward from the top of the screen. This is the gesture used to close a running Metro application. If you’re not running on a touch screen, you can press Alt-F4 instead.
  8. Return to Visual Studio and wait a few seconds for the process to end. (It generally takes about 10 seconds.)
  9. Press F5 to launch the application again.
  10. Confirm that Contoso Cookbook returns you to the recipe displayed when you closed it.
  11. Go to the preferences page and uncheck the “Remember where I was” checkbox.
  12. Close the application while viewing a recipe page.
  13. Return to Visual Studio and wait for the process to end.
  14. Press F5 to launch the application again.
  15. Confirm that you go to the start page, and *not* to the recipe you last viewed.
  16. Return to Visual Studio and stop debugging.

Summary

* 1. Settings and preferences are an important part of virtually every Metro application. Metro’s Settings charm provides a familiar and consistent model for viewing and editing application settings, and as you learned in this lab, it’s simple for a Metro-style app app to expose content through the settings pane.
  2. We’ve come a long way since Lab 1, but there is still more to do. Next up is another important step on the road to Metro stardom: tiles and notifications.