Hands-On Lab

Creating a Windows 8 Metro Style App

Lab version: 1.0.0

Last updated: 6/9/2012



Contents

[Overview 3](#_Toc321032925)

[Exercise 1: Create a Metro style app 4](#_Toc321032926)

[Task 1 – Create the Project 4](#_Toc321032927)

[Task 2 – Familiarize Yourself with the Project 7](#_Toc321032928)

[Task 3 – Customize the Start Page 8](#_Toc321032929)

[Task 4 – Customize the Branding 8](#_Toc321032930)

[Exercise 2: Load Recipe Data 10](#_Toc321032931)

[Task 1 – Prepare Local Recipe Data 10](#_Toc321032932)

[Task 2 – Load Recipe Data 11](#_Toc321032933)

[Exercise 3: Customize the UI 13](#_Toc321032934)

[Task 1 – Modify the Start Page 13](#_Toc321032935)

[Task 2 – Modify the Group-Detail Page 14](#_Toc321032936)

[Task 3 – Modify the Item-Detail Page 16](#_Toc321032937)

[Summary 20](#_Toc321032938)

Overview

* 1. Contoso Cookbook is a series of hands on labs designed to immerse you into Windows 8 Metro style app development. As you work though the labs, you will create a beautiful, functional, real world, Metro style app that leverages some of the key features available in Windows 8. By the end of the series, you will know how to build an application that incorporates many of the key characteristics of a great Metro style app, including:
* A Metro user experience that leverages the signature Windows 8 controls such as GridView, ListView, FlipView, AppBar, and Semantic Zoom
* A user experience that scales across large and small displays and provides proper handling of snapping and different orientations
* Integration with Windows 8 charms through the settings, search, and share contracts
* Handling of lifecycle and application-model events to properly save and restore state and roam settings so the users can seamless transition across tasks and even devices
* Seamless integration with modern hardware to implement features such as photo and video capture
* Secondary tile pinning, notifications and badges to keep your application’s content alive ever-present to the end-user
* Integration with the Windows store APIs for trial and in-app purchasing

In this first lab in the series, you will use XAML and C# to create the application, implement navigation, download the data from Windows Azure (or load it locally if you don’t have an Internet connection), and connect the data to controls using data binding.

# Objectives

* 1. This lab will show you how to:
  + Create a new Metro style app using Visual Studio templates
  + Understand the structure of the project and the files included
  + Brand the application by supplying custom imagery for tiles and other elements
  + Use HttpClient class to retrieve recipe data from Windows Azure
  + Consume that data and data-bind to a GridView control
  + Use data templates to customize the way data is presented by a ListView
  + Modify the code and markup generated by Visual Studio to customize your application’s UI

# 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

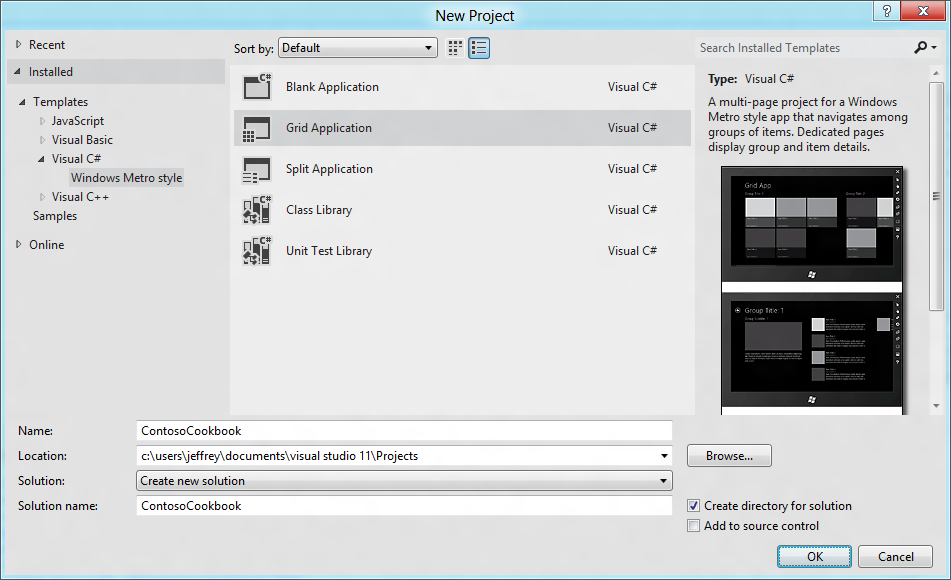
# Exercises

* 1. This Hands-On Lab comprises the following exercises:
  2. Create a Metro style app
  3. Load Recipe Data
  4. Customize the UI
  5. Estimated time to complete this lab:  **40 to 60 minutes**.

Exercise 1: Create a Metro style app

1. In the first exercise, you’ll create a new solution in Visual Studio containing a C# Windows Metro style Grid Application project. Then you’ll examine the files Visual Studio generated and make some simple modifications to customize the application’s UI.

Task 1 – Create the Project

* 1. The first step is to create a new project to house the code and resources that will make up the Contoso Cookbook application, and to see what Visual Studio includes in that project.
  2. Start Visual Studio and use the **File - New Project** command to create a new Visual C# project named “ContosoCookbook.” Be sure to select “Windows Metro Style” from the list of Visual C# templates, and to select “Grid Application” from the list of template types, as shown in Figure 1.
  3. 
  4. Figure 1
  5. Creating the ContosoCookbook project
  6. Select **Start Debugging** from the **Debug** menu (or simply press F5) to launch the application in the debugger. The application will start and you’ll see the screen shown in Figure 2. This is the application’s home page or *start page*.
  7. 
  8. Figure 2
  9. The Contoso Cookbook start page
  10. Take a few moments to play with the application. For starters, use the mouse (or a finger if you’re running on a touch screen) to scroll the screen horizontally.
      1. **Note:** The horizontal scrolling is provided by a ScrollViewer control, and layout is provided by a GridView control. These are but a few of the controls provided to XAML developers in WinRT’s Windows.UI.Xaml.Controls namespace for building rich, compelling UIs.
  11. Find out what happens if you touch or click on one of the GridView items. For example, tap the item labeled “Item Title: 1” to display the screen shown in Figure 3. This is the *item-detail page*.
  12. **Note:**  Windows 8 is often described as a “touch-first” operating system, but it has great support for traditional input devices such as mice and styluses as well. From this point forward, when instructed to “touch” or “tap“ something on the screen, realize that you don’t have to have a touch screen to do it. A simple mouse click will do!
  13. 
  14. Figure 3
  15. The item-detail page
      1. **Note:** When you’re on the item-detail page, you can scroll horizontally to view all the items in the group. That scrolling is provided by a FlipView control, which is yet another of the cool controls featured in Windows.UI.Xaml.Controls.
  16. Go back to the application’s start page by tapping the back button (the circled left-arrow) in the upper-left corner of the screen.
  17. Tap “Group Title: 1” under “ContosoCookbook” in the upper-left corner of the start page to display the *group-detail page* (Figure 4).
  18. 
  19. Figure 4
  20. The group-detail page
  21. Switch back to Visual Studio (if you’re using a touch screen, the easy way to do it is to swipe from left to right starting at the left edge of the screen; if you prefer using the keyboard, press the Windows key and D, or Win-D) and select **Stop Debugging** from the **Debug** menu to stop the application.

Task 2 – Familiarize Yourself with the Project

* 1. It’s clear that when Visual Studio generated the project, it gave you a lot for free. Specifically, it gave you several XAML pages, logic and UI for navigating between pages (including working back buttons), and sample data resources. To implement Contoso Cookbook, we’ll build on what Visual Studio generated. But first take a moment to familiarize yourself with the project structure and with the assets Visual Studio created.
  2. In the Solution Explorer window, check out the contents of the project’s root folder. You’ll find four key files there, plus code-behind files to go with them:
  + App.xaml, which represents the application and its resources
  + GroupedItemsPage.xaml, which represents the start page
  + ItemDetailPage.xaml, which represents the item-detail page
  + GroupDetailPage.xaml, which represents the group-detail page
  1. Look in the project’s Assets folder, where you’ll find image assets used to brand the app.
  2. Look in the project’s Common folder. Among the files you’ll find there are BooleanToVisibilityConverter.cs, which contains a value converter that converts the Boolean values true and false into Visibility.Visible and Visibility.Collapsed, and a file named StandardStyles.xaml, which contains XAML resources used to style the application.
  3. Look in the project’s DataModel folder, where you’ll find a file named SampleDataSource.cs containing data classes as well as sample data to go with them.

Task 3 – Customize the Start Page

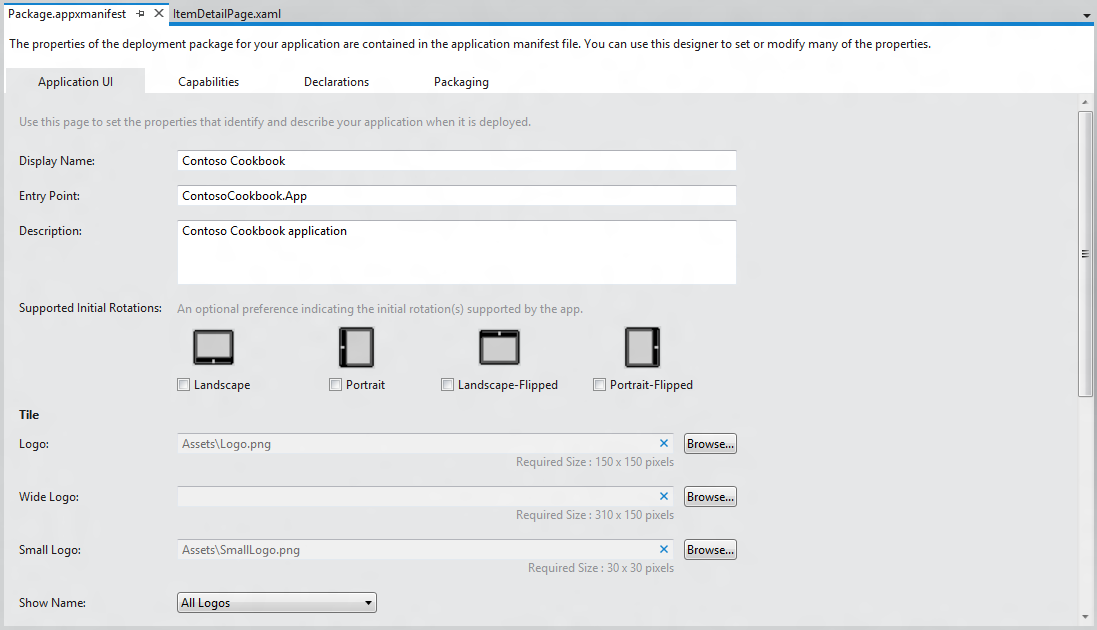
* 1. Currently, the project name – ContosoCookbook – appears at the top of the start page. Let’s modify that to read “Contoso Cookbook.”
  2. Open App.xaml in Visual Studio.
  3. Find the string resource named “AppName” and change its value from “ContosoCookbook” to “Contoso Cookbook,” as shown below:
  4. XAML

<x:String x:Key="AppName">Contoso Cookbook</x:String>

* 1. Press F5 to launch the application in the debugger and confirm that the title text at the top of the start page has changed (Figure 5).
  2. 
  3. Figure 5
  4. The modified start page
  5. Return to Visual Studio and use the **Stop Debugging** command to close the application.

Task 4 – Customize the Branding

If you go out to the Windows 8 start screen right now, you’ll see that there’s a ContosoCookbook tile. That tile is the application’s *primary tile*. It was created when the app was installed, which happened the first time the app was launched from Visual Studio. The image on the tile comes from Logo.png in the Assets folder. In this task, you’ll replace the logo that Visual Studio generated with one more suitable for an electronic cookbook. While you’re at it, you’ll replace the other PNGs in the Assets folder to uniquely brand the application, and finish up by modifying the application manifest.

* 1. On the Windows 8 start screen, right-click the ContosoCookbook tile (or use a finger to drag it down a half inch or so before letting go) and select “Uninstall” to uninstall the application and remove the tile.
  2. Go back to Visual Studio and right-click the Assets folder. Then use the **Add - Existing Item** command to import logo.png, smalllogo.png, splashscreen.png, and storelogo.png from the images folder of the lab starting materials. When prompted, allow these files to overwrite the existing files with the same names.
  3. In Solution Explorer, double-click Package.appxmanifest to open the application manifest.
     1. **Note:** The application manifest contains metadata regarding a Metro style app and is embedded in every application that you build. At runtime, the manifest tells Windows 8 everything it needs to know about the app, including the application name, publisher, and what “capabilities” the application requires, including access to webcams, microphones, the Internet, and parts of the file system – specifically, the user’s documents, music, and videos libraries.
  4. Change the application’s display name to “Contoso Cookbook” and its description to “Contoso Cookbook application,” as shown in Figure 6.
  5. 
  6. Figure 6
  7. Changing the display name and description in the manifest
  8. Press F5 to launch the application.
  9. Watch as the application starts up. Is the splash screen (the screen that’s briefly shown as the app loads) different than before?
  10. Go to the Window 8 start screen and confirm that it contains a tile like the one below.
  11. 
  12. Figure 7
  13. The new application tile
  14. Return to Visual Studio and stop debugging.

Exercise 2: Load Recipe Data

1. 샘플 테이터 말고 실제 레시피 데이터를 가져와서 표시해봅시다. 우리 강의장은 현재 인터넷이 되는 상태기 때문에 Task 1은 넘어가도 됩니다. 우리는 Task 2 부터 진행하고 원격 서버에 있는 레시피 데이터를 사용해보죠. 그래야 비동기가 실감이 나니까요.

~~Task 1 – Prepare Local Recipe Data~~

* 1. ~~If you don’t have an Internet connection or would prefer for the app to use local data rather than data from the cloud, perform the following steps. If you’d rather fetch recipe data from a preexisting Azure service, then you can~~ **~~skip Task 1~~** ~~and go straight to Task 2.~~
  2. ~~Create a new folder named “Data” in the ContosoCookbook project.~~
  3. ~~Right-click the Data folder and use the~~ **~~Add - Existing Item~~** ~~command to import Recipes.txt from the data folder of the starting materials.~~
  4. **~~Note:~~** ~~If you take a moment to look inside Recipes.txt, you’ll see that it contains nothing more than JSON-encoded data denoting recipes and recipe groups. The data in the file is identical to the data retrieved from Azure except for the image URLs. We’ll be using classes in WinRT’s Windows.Data.Json namespace to consume this data.~~
  5. ~~Add a folder named “Images” to the project.~~
  6. ~~Import the folders named chinese, french, german, indian, italian, and mexican (along with their contents) from the images folder of the starting materials to the project’s Images folder. It’s important to put them in the Images folder, because the image URLs in Recipes.txt assume that’s where they’re located.~~
     1. **~~Note:~~** ~~An easy way to do the import is to drag the folders from a window open on the desktop and drop them over the Images folder in Solution Explorer.~~

Task 2 – Load Recipe Data

* 1. The next step is to replace the sample data that Visual Studio provided with code that loads recipe data. **Perform this task even if you performed Task 1**.
  2. Right-click the DataModel folder in Solution Explorer and use the **Add - Existing Item** command to import RecipeDataSource.cs from the data folder of the starting materials.
     1. **Note:** Visual Studio 의 Grid Application 템플릿은 기본적으로 SampleDataSource.cs 라는 파일에 총4개의 클래스(SampleDataCommon, SampleDataItem, SampleDataGroup, SampleDataSource)를 통해서 데이터를 가져오는 샘플을 제공합니다. 코드가 쉽지는 않지만 잘 이해를 하면 큰 도움이 됩니다. 이런식으로 데이터를 가져오는 부분(Data Model)을 따로 빼서 잘 만들어 두면 데이터 소스가 변경되어도 문제없이 수정할 수 있습니다.
  3. App.xaml.cs 를 열어서 제일 상단에 using 문을 넣어준다.
     1. C#

using ContosoCookbook.Data;

* 1. App.xaml.cs 에서 앱의 시작점은OnLaunched 메서드 입니다. rootFrame.Navigate 메서드가 “AllGroups”라는 파라미터와 함께GroupedItemsPage 를 호출해 줍니다. OnLaunched 메서드에서 앱에서 사용할 데이터를 비동기로 가져옵니다.
     1. C#
  2. protected override async void OnLaunched(LaunchActivatedEventArgs args)
  3. {
  4. // Do not repeat app initialization when already running, just ensure that
  5. // the window is active
  6. if (args.PreviousExecutionState == ApplicationExecutionState.Running)
  7. {
  8. Window.Current.Activate();
  9. return;
  10. }
  11. // Create a Frame to act as the navigation context and associate it with
  12. // a SuspensionManager key
  13. var rootFrame = new Frame();
  14. SuspensionManager.RegisterFrame(rootFrame, "AppFrame");
  15. await RecipeDataSource.LoadRemoteDataAsync();
  16. if (args.PreviousExecutionState == ApplicationExecutionState.Terminated)
  17. {
  18. // Restore the saved session state only when appropriate
  19. await SuspensionManager.RestoreAsync();
  20. }
  21. if (rootFrame.Content == null)
  22. {
  23. // When the navigation stack isn't restored navigate to the first page,
  24. // configuring the new page by passing required information as a navigation
  25. // parameter
  26. if (!rootFrame.Navigate(typeof(GroupedItemsPage), "AllGroups"))
  27. {
  28. throw new Exception("Failed to create initial page");
  29. }
  30. }
  31. // Place the frame in the current Window and ensure that it is active
  32. Window.Current.Content = rootFrame;
  33. Window.Current.Activate();
  34. }
  35. GroupedItemsPage.xaml.cs 를 열어LoadState 메서드를 찾아서 기존 샘플 데이터를 가져오는 부분을 지우고 레시피 데이터를 불러오는 부분을 넣습니다.
      1. C#
  36. ~~var sampleDataGroups = SampleDataSource.GetGroups((String)navigationParameter);~~
  37. ~~this.DefaultViewModel["Groups"] = sampleDataGroups;~~
  38. var recipeDataGroups = RecipeDataSource.GetGroups((String)navigationParameter);

this.DefaultViewModel["Groups"] = recipeDataGroups;

* 1. GroupedItemsPage.xaml.cs 에서 그룹을 누를때(Header\_Click 메서드) 이벤트 핸들러에서도 샘플 데이터의 흔적을 지워줍니다.
     1. C#
  2. ~~this.Frame.Navigate(typeof(GroupDetailPage), ((SampleDataGroup)group).UniqueId);~~

this.Frame.Navigate(typeof(GroupDetailPage), ((RecipeDataGroup)group).UniqueId);

* 1. GroupedItemsPage.xaml.cs 에서아이템을 눌렀을 때(ItemView\_ItemClick 메서드)의 이벤트 핸들러에서도 샘플 데이터의 흔적을 지워줍니다.
     1. C#

~~var itemId = ((SampleDataItem)e.ClickedItem).UniqueId;;~~

var itemId = ((RecipeDataItem)e.ClickedItem).UniqueId;

* 1. GroupedItemsPage.xaml을 열어서 샘플데이터 사용 부분을 레시피 데이터로 변경해 줍니다. :
     1. C#
     2. ~~<Page.Resources>~~
     3. ~~<CollectionViewSource~~
     4. ~~x:Name="groupedItemsViewSource"~~
     5. ~~Source="{Binding Groups}"~~
     6. ~~IsSourceGrouped="true"~~
     7. ~~ItemsPath="TopItems"~~
     8. ~~d:Source="{Binding AllGroups, Source={d:DesignInstance Type=data: SampleDataSource, IsDesignTimeCreatable=True}}"/>~~
     9. ~~</Page.Resources>~~
  2. <Page.Resources>
  3. <CollectionViewSource
  4. x:Name="groupedItemsViewSource"
  5. Source="{Binding Groups}"
  6. IsSourceGrouped="true"
  7. ItemsPath="TopItems"
  8. d:Source="{Binding AllGroups, Source={d:DesignInstance Type=data:RecipeDataSource, IsDesignTimeCreatable=True}}"/>
  9. </Page.Resources>
  10. 그룹이름을 눌렀을 때 이동하는 페이지 인GroupDetailPage.xaml.cs 를 열어LoadState부분에서 샘플 데이터를 레시피 데이터로 변경해 줍니다. :
      1. C#
  11. ~~var group = SampleDataSource.GetGroup((String)navigationParameter);~~
  12. var group = RecipeDataSource.GetGroup((String)navigationParameter);
  13. GroupDetailPage.xaml.cs 에서 레시피 아이템을 눌렀을 때 발생하는 이벤트 핸들러인ItemView\_ItemClick 메서드를 찾아 샘플 데이터의 흔적을 레시피데이터로 변경합니다.:
      1. C#
  14. ~~var itemId = ((SampleDataItem)e.ClickedItem).UniqueId;~~

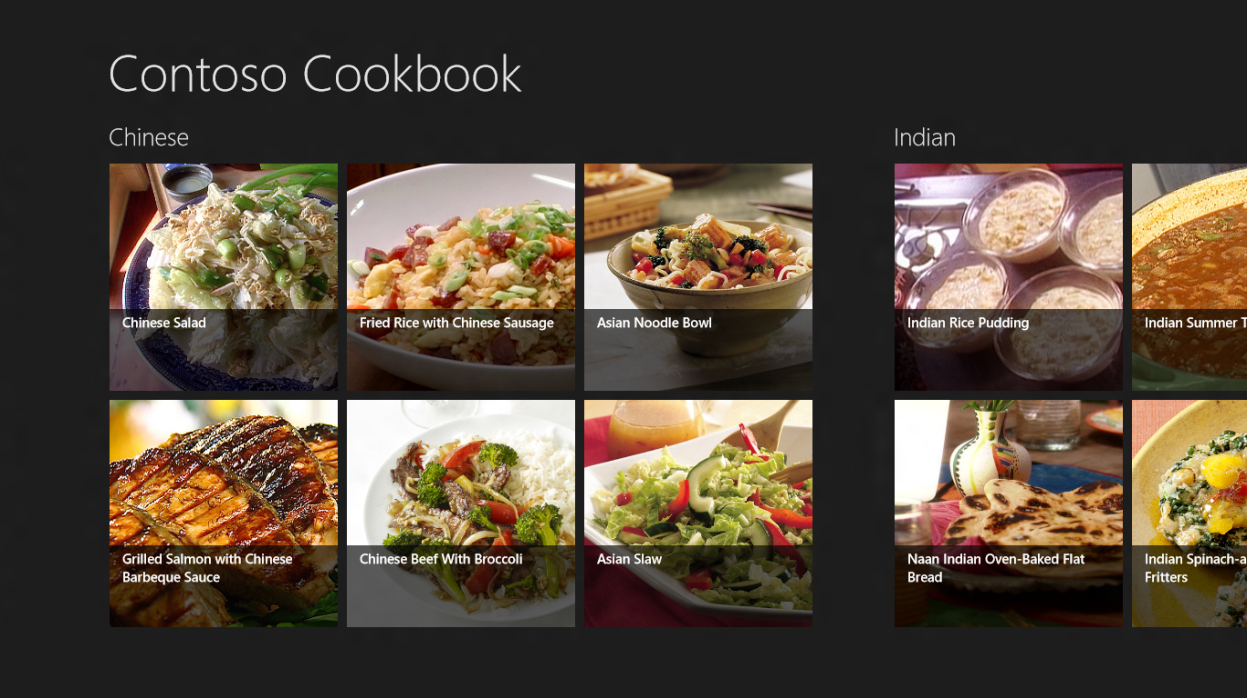
var itemId = ((RecipeDataItem)e.ClickedItem).UniqueId;

* 1. GroupDetailPage.xaml 를 열어서 샘플데이터의 흔적을 레시피데이터로 변경합니다.
     1. C#
  2. ~~<Page.Resources>~~
  3. ~~<!-- Collection of items displayed by this page -->~~
  4. ~~<CollectionViewSource~~
  5. ~~x:Name="itemsViewSource"~~
  6. ~~Source="{Binding Items}"~~
  7. ~~d:Source="{Binding AllGroups[0].Items, Source={d:DesignInstance Type=data:SampleDataSource, IsDesignTimeCreatable=True}}"/>~~

~~</Page.Resources>~~

* 1. <Page.Resources>
  2. <!-- Collection of items displayed by this page -->
  3. <CollectionViewSource
  4. x:Name="itemsViewSource"
  5. Source="{Binding Items}"
  6. d:Source="{Binding AllGroups[0].Items, Source={d:DesignInstance Type=data:RecipeDataSource, IsDesignTimeCreatable=True}}"/>

</Page.Resources>

* 1. 레시피 상세 페이지인 ItemDetailPage.xaml.cs 를 열어서 페이지의 시작 부분인LoadState 메서드를 찾아서 샘플데이터 사용부분을 레시피 데이터로 변경합니다. :
     1. C#
  2. ~~var item = SampleDataSource.GetItem((String)navigationParameter~~);
  3. var item = RecipeDataSource.GetItem((String)navigationParameter);
  4. ItemDetailPage.xaml.cs 에서SaveState 메서드에서 사용한 샘플데이터를 레시피 데이터로 변경합니다.
     1. C#
  5. ~~var selectedItem = (SampleDataItem)this.flipView.SelectedItem;~~
  6. var selectedItem = (RecipeDataItem)this.flipView.SelectedItem;
  7. Press F5 to debug the application and verify that the start page looks like the one below.
  8. 
  9. Figure 8
  10. The start page with recipes
  11. Return to Visual Studio and stop debugging.

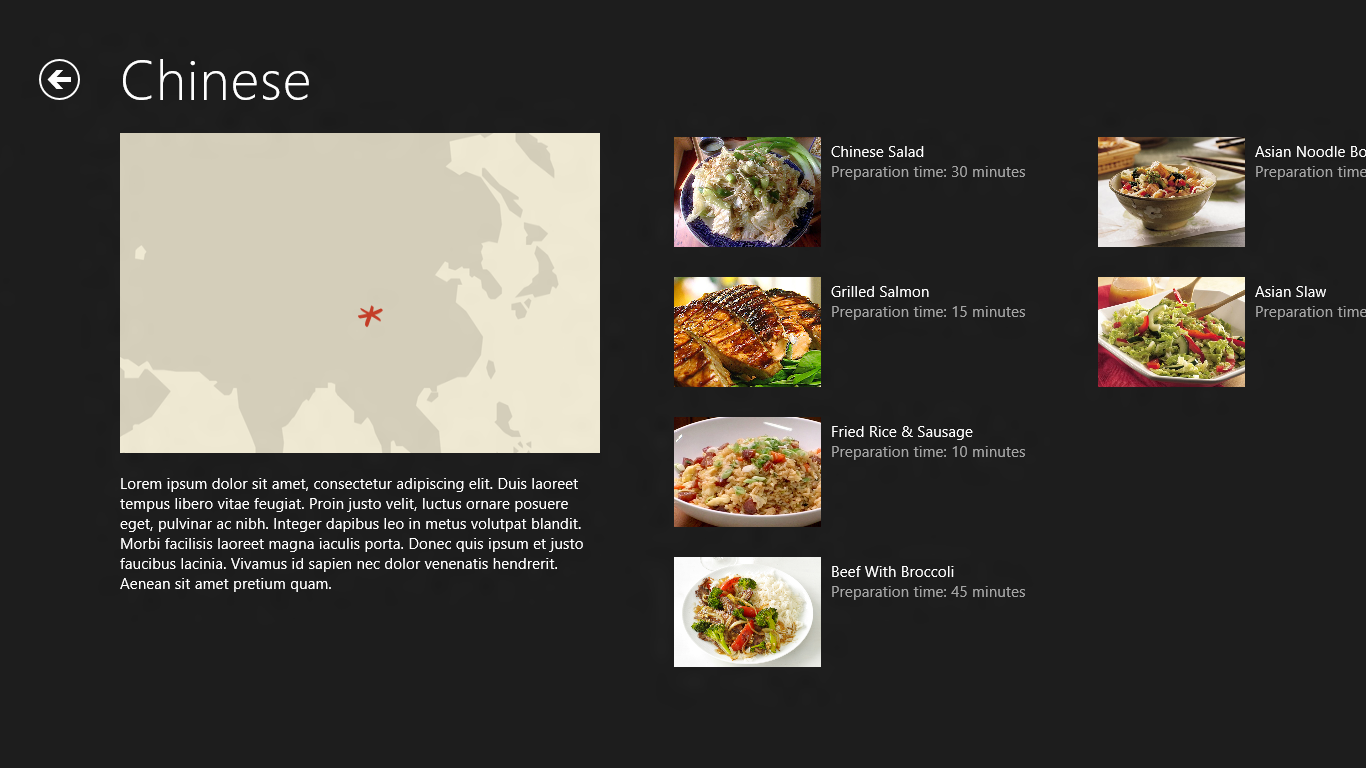
Exercise 3: Customize the UI

* 1. That’s a great start considering that we’ve written precious little code so far, but we need to customize the UI and mold it to our domain-specific data model. In this exercise, you’ll modify the start page, the item-detail page, and the group-detail page to refine the look of Contoso Cookbook.

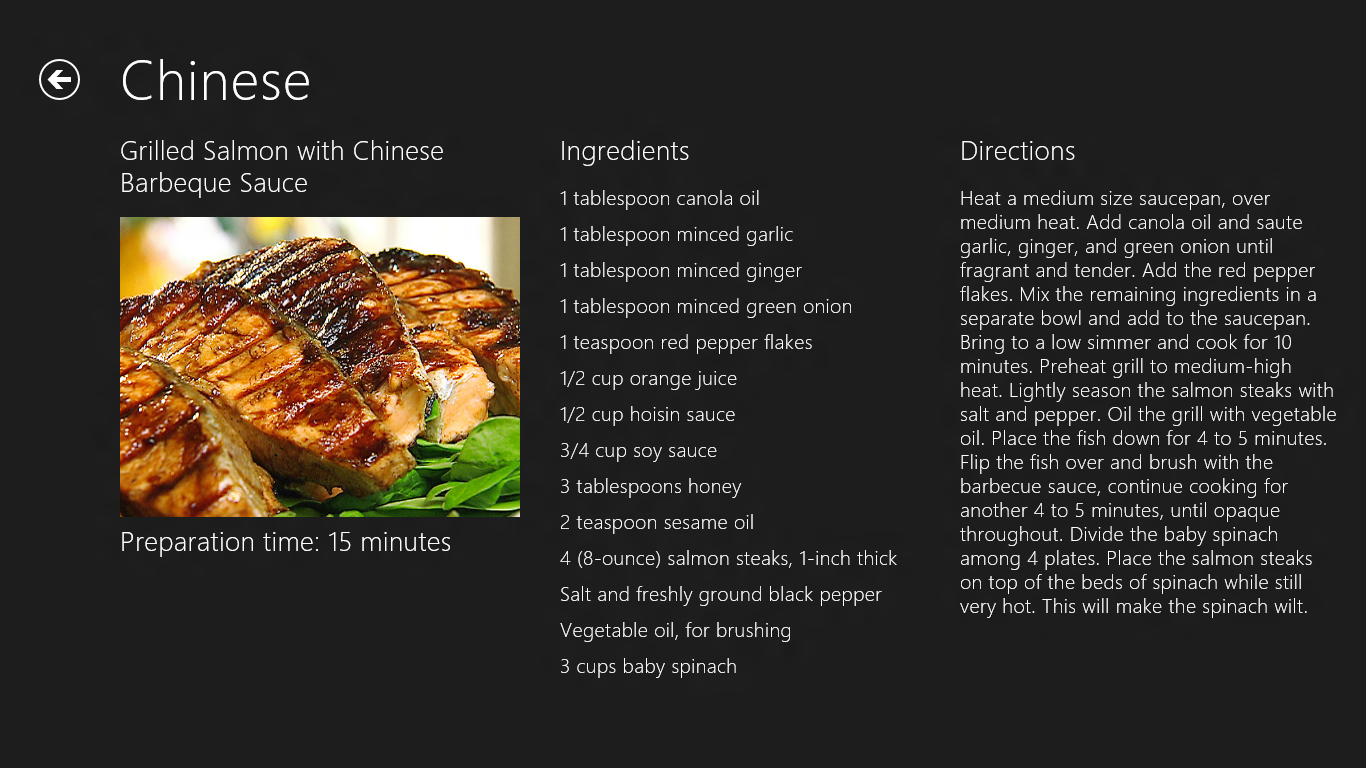
Task 1 – Modify the Start Page

* 1. Let’s begin by modifying the start page to improve the look of the recipe items.
  2. Open StandardStyles.xaml, which is located in the project’s Common folder.
  3. Find the DataTemplate element whose name is “Standard250x250ItemTemplate” (it begins at line 823). This is the date template used to render recipe items on the start page.
  4. Remove the final TextBlock in the data template (the TextBlock whose Text property is bound to “Subtitle”) since the RecipeDataItem class doesn’t have a Subtitle property.
  5. In the same data template, change the width and height of the root Grid to 320 by 240 to preserve the original aspect ratio of the recipe images. Also change the height of the remaining TextBlock from 60 to 48 to decrease the height of the partially transparent black overlay at the bottom of each item, and change the property bound to the TextBlock’s Text property from “Title” to “ShortTitle”:
     1. XAML
     2. <DataTemplate x:Key="Standard250x250ItemTemplate">
     3. <Grid HorizontalAlignment="Left" Width="320" Height="240">
     4. <Border Background="{StaticResource ListViewItemPlaceholderRectBrush}">
     5. <Image Source="{Binding Image}" Stretch="UniformToFill"/>
     6. </Border>
     7. <StackPanel VerticalAlignment="Bottom" Background="{StaticResource ListViewItemOverlayBackgroundBrush}">
     8. <TextBlock Text="{Binding ShortTitle}" Foreground="{StaticResource ListViewItemOverlayTextBrush}" Style="{StaticResource TitleTextStyle}" Height="48" Margin="15,0,15,0"/>
     9. </StackPanel>
     10. </Grid>
     11. </DataTemplate>
  6. Now press F5 to run the application. Confirm that the recipe items on the start page look like the ones below.
  7. 
  8. Figure 9
  9. The new and improved start page
  10. Return to Visual Studio and stop debugging.

Task 2 – Modify the Group-Detail Page

* 1. You’ve modified the start page to improve the look of the app, but you also need to modify the group-detail page. In this task, you’ll revise that page to make group details more presentable.
  2. Start the application again and tap “Chinese” in the upper-left corner of the screen to navigate to the group-detail page showing Chinese recipes. The changes we’ll make here are minor. We’ll close up some of the space between “Chinese” and the image below it, we’ll replace recipe titles with short titles, and we’ll add a preparation time to each recipe.
  3. Return to Visual Studio and stop debugging.
  4. Open GroupDetailPage.xaml “itemGridView” 이라는 이름의GridView 를 찾아서 (<GridView x:Name="itemGridView" …)
  5. 아래쪽 마진을 10으로 변경시켜준다.
     1. ~~XAML~~
  6. <GridView
  7. x:Name="itemGridView"
  8. AutomationProperties.AutomationId="ItemGridView"
  9. AutomationProperties.Name="Items In Group"
  10. TabIndex="1"
  11. Grid.Row="1"
  12. Margin="0,-14,0,10"
  13. Padding="120,0,120,50"
  14. “itemGridView” 이라는 이름의GridView 를 찾아서 (<GridView x:Name="itemGridView" …) <GridView.Header> 밑에 StackPanel을 찾는다.
  15. Remove the first TextBlock from the StackPanel.
  16. In the Image element on the next line, replace Height=”400” with Width=”480” and add a HorizontalAlignment=”Left” attribute:
      1. XAML
      2. <Image Source="{Binding Image}" Width="480" Margin="0,0,18,20" Stretch="UniformToFill" HorizontalAlignment="Left"/>
  17. Now go back to StandardStyles.xaml and find the DataTemplate element whose name is “Standard500x130ItemTemplate.” This is the data template used to render recipe items on the group-detail page.
  18. Change the width of the root Grid in the data template from 480 to 360:
      1. XAML
      2. <Grid Height="110" Width="360" Margin="10">
  19. Change the width of the Border inside the data template from 110 to 147 to preserve the aspect ratios of the recipe images:
      1. XAML
      2. <Border Background="{StaticResource ListViewItemPlaceholderRectBrush}" Width="147" Height="110">
  20. Just below that, remove the TextBlocks whose Text properties are bound to the data source’s “Subtitle” and “Description” properties.
  21. In the remaining TextBlock, change the property bound to the Text property from “Title” to “ShortTitle:”
      1. XAML
      2. <TextBlock Text="{Binding ShortTitle}" Style="{StaticResource TitleTextStyle}" TextWrapping="NoWrap"/>
  22. Underneath that TextBlock, add the following statements to include a preparation time below the recipe title:
      1. XAML
      2. <StackPanel Orientation="Horizontal">
      3. <TextBlock Text="Preparation time:" Style="{StaticResource BodyTextStyle}" />
      4. <TextBlock Text="{Binding PrepTime}" Style="{StaticResource BodyTextStyle}" Margin="4,0,4,0" />
      5. <TextBlock Text="minutes" Style="{StaticResource BodyTextStyle}" />
      6. </StackPanel>
  23. When you’re done, here’s what the modified data template should look like:
      1. XAML
      2. <DataTemplate x:Key="Standard500x130ItemTemplate">
      3. <Grid Height="110" Width="360" Margin="10">
      4. <Grid.ColumnDefinitions>
      5. <ColumnDefinition Width="Auto"/>
      6. <ColumnDefinition Width="\*"/>
      7. </Grid.ColumnDefinitions>
      8. <Border Background="{StaticResource ListViewItemPlaceholderRectBrush}" Width="147" Height="110">
      9. <Image Source="{Binding Image}" Stretch="UniformToFill"/>
      10. </Border>
      11. <StackPanel Grid.Column="1" VerticalAlignment="Top" Margin="10,0,0,0">
      12. <TextBlock Text="{Binding ShortTitle}" Style="{StaticResource TitleTextStyle}" TextWrapping="NoWrap"/>
      13. <StackPanel Orientation="Horizontal">
      14. <TextBlock Text="Preparation time:" Style="{StaticResource BodyTextStyle}" />
      15. <TextBlock Text="{Binding PrepTime}" Style="{StaticResource BodyTextStyle}" Margin="4,0,4,0" />
      16. <TextBlock Text="minutes" Style="{StaticResource BodyTextStyle}" />
      17. </StackPanel>
      18. </StackPanel>
      19. </Grid>
      20. </DataTemplate>
  24. Start the application and tap “Chinese” again. Verify that your group-detail page resembles the one below:
  25. 
  26. Figure 10
  27. The modified group-detail page
  28. Return to Visual Studio and stop debugging.

Task 3 – Modify the Item-Detail Page

* 1. The final task in crafting a basic UI for the application is to modify the item-detail page to present more information about recipes, including directions and ingredients.
  2. Run the application and tap “Steam Bun baos” in the upper-left corner. Clearly, we have some work to do on the item-detail page.
  3. Return to Visual Studio and stop debugging.
  4. Right-click the Common folder in Solution Explorer and use the **Add - New Item** command to add a new class to the project. Name the file ListConverter.cs.
  5. Replace the file’s contents with this:
     1. C#
     2. using System;
     3. using System.Collections.Generic;
     4. using System.Collections.ObjectModel;
     5. using System.Linq;
     6. using System.Text;
     7. using System.Threading.Tasks;
     8. using Windows.UI.Xaml.Data;
     9. namespace ContosoCookbook.Common
     10. {
     11. class ListConverter : IValueConverter
     12. {
     13. public object Convert(object value, Type targetType, object parameter, string language)
     14. {
     15. ObservableCollection<string> items = (ObservableCollection<string>) value;
     16. StringBuilder builder = new StringBuilder();
     17. foreach(var item in items)
     18. {
     19. builder.Append(item);
     20. builder.Append("\r\n");
     21. }
     22. return builder.ToString();
     23. }
     24. public object ConvertBack(object value, Type targetType, object parameter, string language)
     25. {
     26. throw new NotImplementedException();
     27. }
     28. }
     29. }
     30. **Note:** ListConverter is a value converter that converts an array of strings into a single string containing line breaks. We need it because we’ll be binding the Text property of a TextBlock to an array of strings, and that requires a value converter.
  6. Open ItemDetailPage.xaml and add the following statement to the <Page.Resources> section near the top of the file to declare a ListConverter instance:
     1. XAML
     2. <common:ListConverter x:Key="ListConverter" />
  7. Scroll down in ItemDetailPage.xaml and find the FlipView whose name is “flipView.”
  8. Replace the DataTemplate element inside the FlipView with this one:
     1. XAML
     2. <DataTemplate>
     3. <UserControl Loaded="StartLayoutUpdates" Unloaded="StopLayoutUpdates">
     4. <ScrollViewer x:Name="scrollViewer" Style="{StaticResource VerticalScrollViewerStyle}" Grid.Row="1">
     6. <!-- Three-column grid for item-detail layout -->
     7. <Grid Margin="120,0,0,0">
     8. <Grid.ColumnDefinitions>
     9. <ColumnDefinition Width="400" />
     10. <ColumnDefinition Width="40" />
     11. <ColumnDefinition Width="360" />
     12. <ColumnDefinition Width="40" />
     13. <ColumnDefinition Width="360" />
     14. </Grid.ColumnDefinitions>
     16. <StackPanel Orientation="Vertical" Grid.Column="0">
     17. <TextBlock FontSize="26.667" FontWeight="Light" Text="{Binding Title}" TextWrapping="Wrap"/>
     18. <Image x:Name="image" Width="400" Margin="0,20,0,10" Stretch="Uniform" Source="{Binding Image}"/>
     19. <StackPanel Orientation="Horizontal">
     20. <TextBlock FontSize="26.667" FontWeight="Light" Text="Preparation time:"/>
     21. <TextBlock FontSize="26.667" FontWeight="Light" Text="{Binding PrepTime}" Margin="8,0,8,0"/>
     22. <TextBlock FontSize="26.667" FontWeight="Light" Text="minutes"/>
     23. </StackPanel>
     24. </StackPanel>
     26. <StackPanel Orientation="Vertical" Grid.Column="2">
     27. <TextBlock FontSize="26.667" FontWeight="Light" Text="Ingredients" Margin="0,0,0,16"/>
     28. <TextBlock FontSize="20" FontWeight="Light" LineHeight="32.5" Text="{Binding Ingredients, Converter={StaticResource ListConverter}}" TextWrapping="Wrap" />
     29. </StackPanel>
     30. <StackPanel Orientation="Vertical" Grid.Column="4">
     31. <TextBlock FontSize="26.667" FontWeight="Light" Text="Directions" Margin="0,0,0,16"/>
     32. <ScrollViewer Style="{StaticResource VerticalScrollViewerStyle}">
     33. <Grid>
     34. <TextBlock FontSize="20" FontWeight="Light" Text="{Binding Directions}" TextWrapping="Wrap" />
     35. </Grid>
     36. </ScrollViewer>
     37. </StackPanel>
     38. </Grid>
     39. </ScrollViewer>
     40. </UserControl>
     41. </DataTemplate>
     42. **Note:** The new data template shows recipes using a 3-column format. The recipe name, image, and preparation time appear in column 1, a list of ingredients appears in column 2, and cooking directions appear in column 3.
  9. Now run the application again. Tap “Grilled Salmon” and verify that the item-detail page looks like the one in Figure 11.
  10. 
  11. Figure 11
  12. The modified item-detail page
  13. Return to Visual Studio and stop debugging.

Summary

* 1. In this lab, you created a new Metro style Grid Applicationproject in Visual Studio, replaced the sample data with real data, replaced the default branding assets with ones tailored to the application, and customized the UI by modifying the some of the XAML provided by Visual Studio. Moreover, you got a first-hand look at how a project is structured and how the pieces fit together.
  2. You also imported code that demonstrates how System.Net.Http.HttpClient can be used to load data from a remote data source and how WinRT’s Windows.Data.Json classes can be used to consume JSON data in C#. By modifying data templates, you customized the way this data is presented to the user.
  3. It’s a great start, but there’s still more to do make Contoso Cookbook a first-class Metro style app. The journey continues in Lab 2!