Windows Store App Development

Lab 1: Creating a Windows Store App

Version 1.6.0

Conditions and Terms of Use

Microsoft Confidential - For Internal Use Only

Contents

Lab 1: Creating a Windows Store App

Introduction

Contoso Cookbook is a series of hands-on labs designed to immerse you in Windows Store app development. As you work though the labs, you will create a beautiful, functional, real-world app that makes use of some of the key new features available in Microsoft Windows. By the end of the series, you will know how to build an app that incorporates many of the key characteristics of a great app for the Windows Store, including:

- A user experience that employs the signature Windows controls such as:
  - **GridView**
  - **ListView**
  - **AppBar**
  - **SemanticZoom**
- A user experience that scales across large and small displays and provides proper handling of different orientations and window sizes.
- Integration with Windows charms through the Settings and Share contracts.
- Handling of lifecycle and app-model events to properly save and restore state and to roam settings so that users can seamlessly move among 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 app's content alive and ever present to users.
- Integration with the Windows Store APIs for trial and in-app purchasing.

In this first lab in the series, you will use Extensible Application Markup Language (XAML) and C# to create the app, implement navigation, download the data from

Windows Azure (or load it locally if you do not have an Internet connection), and connect the data to controls using data binding.

Objectives

This lab will show you how to:

- Create a new Windows Store app using Microsoft Visual Studio 2013 templates.
- Understand the structure of the project.
- Brand the app by supplying custom imagery for tiles and other elements.
- Use the **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** control.
- Modify the code and markup generated by Visual Studio to customize your app's user interface (UI).

System requirements

You must have the following to complete this lab:

- Microsoft Windows 8.1
- Microsoft Visual Studio 2013

Setup

You must perform the following steps to prepare your computer for this lab:

- Install Microsoft Windows 8.1
- Install Microsoft Visual Studio 2013

Exercises

This Hands-on lab includes the following exercises:

1. Creating a Windows Store app
2. Loading recipe data
3. Customizing the UI

Estimated time to complete this lab: **30 to 45 minutes**.

Exercise 1: Creating a Windows Store App

In the first exercise, you will create a new solution in Visual Studio containing a C# Grid App project. Then you will examine the files that Visual Studio generated and make some simple modifications to customize the app's UI.

Task 1 – Create the project

The first step is to create a new project to contain the code and resources that will make up the Contoso Cookbook app, and to see what Visual Studio includes in that project.

1. Start Visual Studio and use the **File > New Project** command to create a new Visual C# project named "ContosoCookbook". Select **Windows Store** from the list of Visual C# templates, and select **Grid App (XAML)** from the list of template types, as shown in Figure 1.

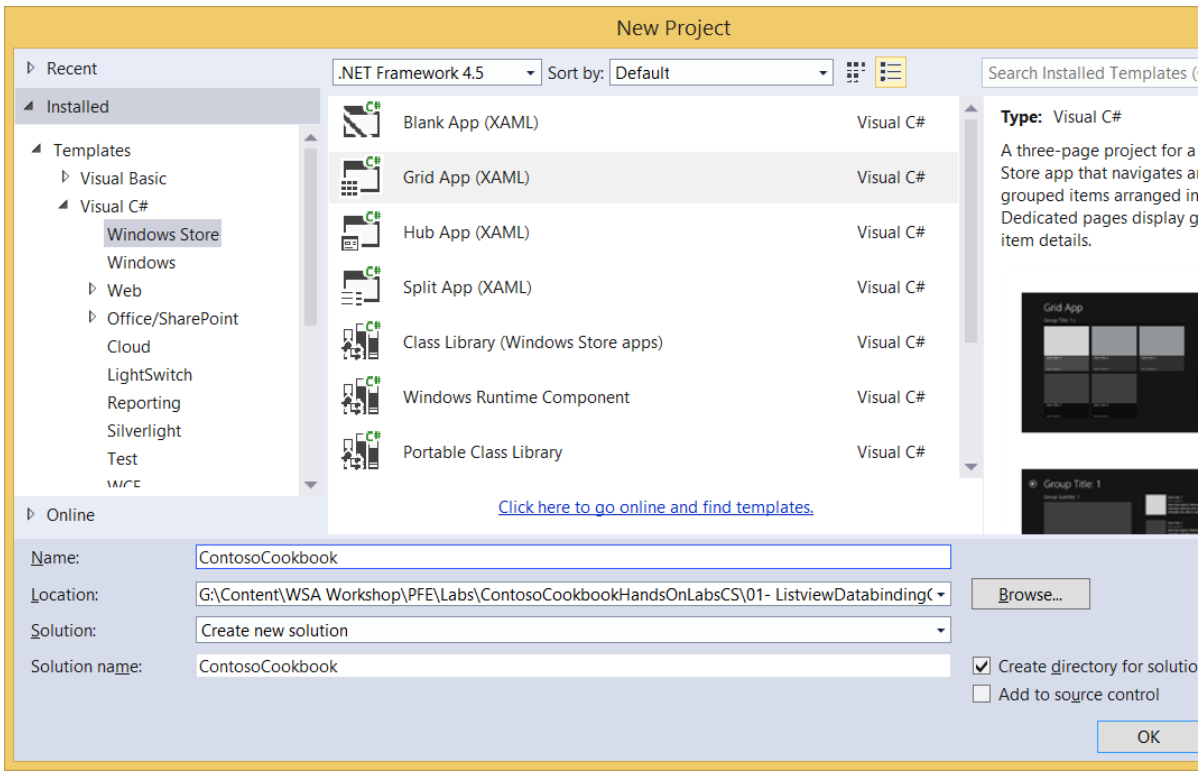

Figure 1: Creating the ContosoCookbook Project

2. Select **Start Debugging** from the **Debug** menu (or press F5) to start the app in the debugger. You will see the screen shown in Figure 2. This is the app's home page or **Start page**.

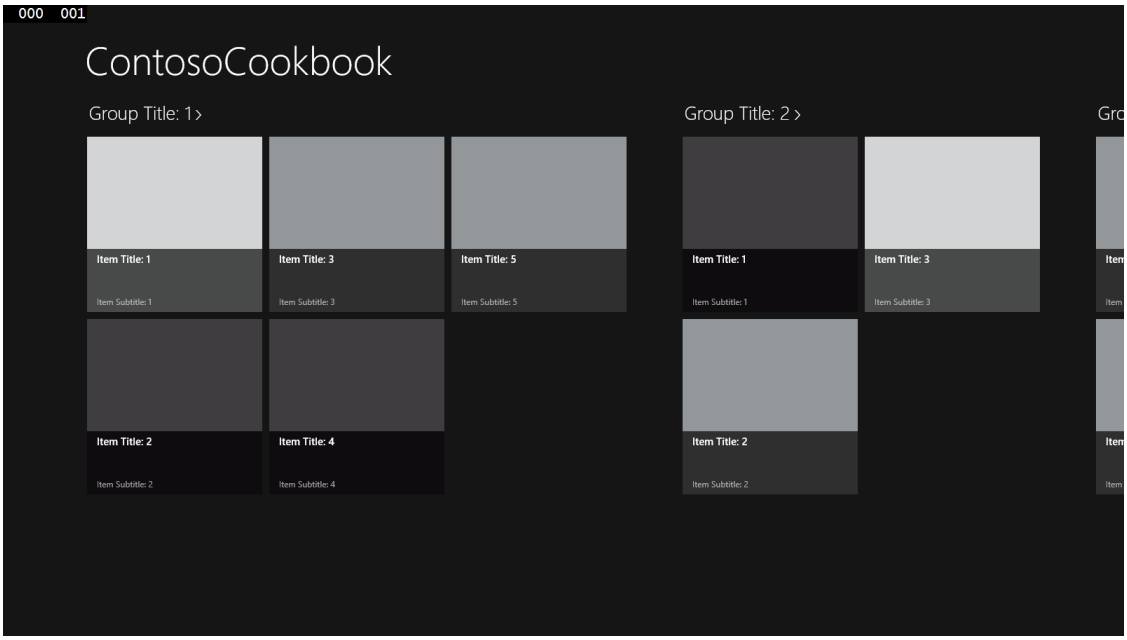

Figure 2: The Contoso Cookbook Start Page

3. Take a few moments to play with the app. For starters, use the mouse (or a finger if your computer has a touch screen) to scroll the screen horizontally.

> **Note:** The grid layout and the horizontal scrolling are provided by a **GridView** control, which is one of many controls provided in the Windows.UI.Xaml.Controls namespace of the Windows Runtime for building rich, compelling UIs.

4. Find out what happens if you touch or click 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**.

> **Note:** Windows is described as a “touch-first” operating system, but it also has great support for traditional input devices such as mice and styluses. From this point forward, when instructed to “touch” or “tap“ something on the screen, realize that you do not have to have a touch screen to do it. A simple mouse click will do.

Figure 3: The Item-Detail Page

> **Note:** When you are on the item-detail page, you can scroll horizontally to view all the items in the group. (If you are using a mouse, click the arrows that appear on the left and right edges of the screen.) That scrolling is provided by a **FlipView** control, which is another of the controls featured in the Windows.UI.Xaml.Controls namespace.[MVDB1]

5. Go back to the app's Start page by tapping the back button (the circled arrow) in the upper-left corner of the screen.
6. Tap **Group Title: 1>** under **ContosoCookbook** in the upper-left corner of the start page to display the **group-detail page** (Figure 4).

Figure 4: The Group-Detail Page

7. Switch back to Visual Studio. (If you are 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 Windows logo key+D.) Then select **Stop Debugging** from the **Debug** menu to stop the app.

Task 2 – Familiarize Yourself with the Project

It is 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 among pages (including working back buttons), and sample data resources. To implement Contoso Cookbook, we will build on what Visual Studio generated. First, take a moment to familiarize yourself with the project structure and with the assets that Visual Studio created.

1. In the Solution Explorer window, check out the contents of the project's root folder. You will find four key files there, plus code-behind files to go with them:
   - **App.xaml**, which represents the app and its resources
   - **GroupedItemsPage.xaml**, which represents the app's start page
   - **ItemDetailPage.xaml**, which represents the item-detail page
   - **GroupDetailPage.xaml**, which represents the group-detail page
2. Look in the project's **Assets** folder, where you will find the image assets used to brand the app.
3. Look in the project's **Common** folder. Among the files, you will find there are **NavigationHelper.cs**, which contains a class that implements code to navigate back and forward between pages, and a file named **suspensionManager.cs**, which

contains a class that handles Process Lifetime Management when the app is suspended.

4. Look in the project's **DataModel** folder, where you will find a file named **SampleDataSource.cs** containing data classes as well as sample data in **SampleData.json**.

Task 3 – Customize the Start Page

Currently, the project name appears at the top of the start page as "ContosoCookbook". Let us modify that to read "Contoso Cookbook."

1. Open **App.xaml** in Visual Studio.
2. Find the string resource named **AppName** and change its value from "ContosoCookbook" to "Contoso Cookbook," as shown here.

XAML

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

3. Press **F5** to start the app in the debugger and confirm that the title text at the top of the start page has changed (Figure 5).

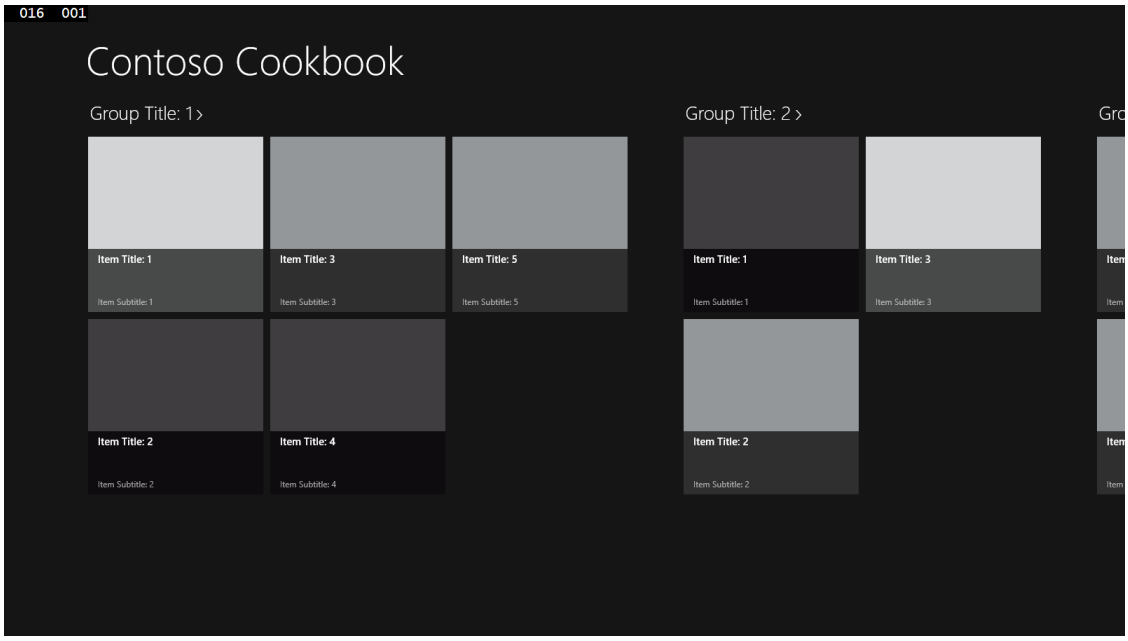

Figure 5: The modified Start Page

4. Return to Visual Studio and use the **Stop Debugging** command to close the app.

Task 4 – Customize the Branding

In this task, you will replace the tile logo that Visual Studio generated with one that is more suitable for a cookbook app. While you are at it, you will replace the other PNG files in the Assets folder to uniquely brand the app, and finish up by modifying the app manifest.

1. Go to the Windows Start screen and navigate to the installed apps by either flicking your finger up or by clicking the arrow that appears at the left bottom of the screen when you move your mouse as in figure 6.

Figure 6

2. In the installed apps screen, right-click the ContosoCookbook tile and click Pin To Start from the Appbar. If you now go out to the Windows Start screen, you will see that there is a "ContosoCookbook" tile. That tile is the app's **primary tile**. It was created when the app was installed, which happened the first time the app was started from Visual Studio. The image on the tile comes from **Logo.png** in the Assets folder.
3. On the Windows 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 app and remove the tile.
4. Go back to Visual Studio and right-click the Assets folder. Then use the **Add > Existing Item** command to import Logo.scale-100.png, SmallLogo.scale-100.png, SplashScreen.scale-100.png, StoreLogo.scale-100.png, and WideLogo.scale-100.png from the Images folder of the lab starting materials. When prompted, allow Visual Studio to write these files over the existing files with the same names.
5. In Solution Explorer, double-click **Package.appxmanifest** to open the app manifest.

> **Note:** The **app manifest**contains the metadata for an app and is embedded in every app that you build. At runtime, the manifest tells Windows everything it needs to know about the app, including the app name, publisher, and what *capabilities* the app requires. Capabilities include access to webcams, microphones, the Internet, and parts of the file system–specifically, the user's pictures, music, and videos libraries.

6. Change the app's display name to "Contoso Cookbook" and its description to "Contoso Cookbook Application", as shown in Figure 6.

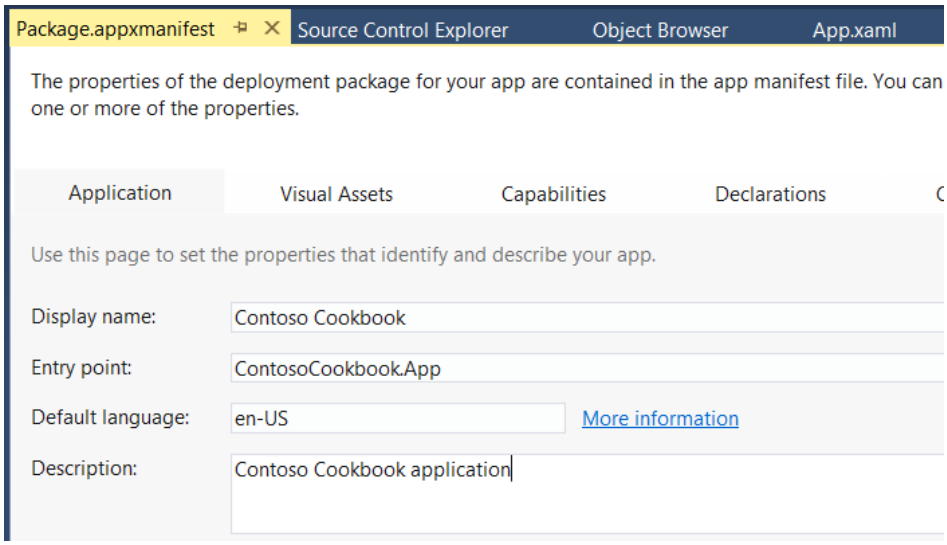

Figure 7: Changing the name in the Manifest

7. In the Visual Assets tab, enter “Assets\WideLogo.scale-100.png” into the **Wide logo** box, to give the app a wide tile.

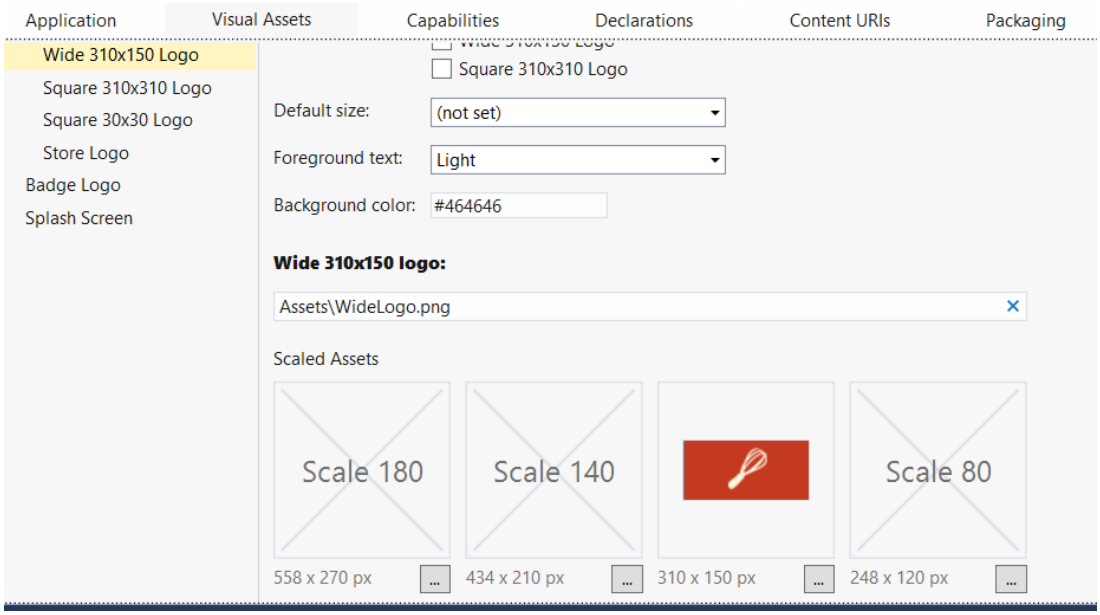

Figure 8: Adding a wide tile

8. Press **F5** to start the app.
9. Watch as the app starts up. Is the splash screen (the screen that is briefly shown as the app loads) different from before?
10. Go to the Windows Start screen and confirm that it contains a tile like this. If you want the name to show up, you need to add a Short Name in the Visual Assets tab, and check the checkboxes for the Tile sizes for which you want to show the name.

Figure 9: The New App Tile

> **Note:** If you would prefer a square tile, right-click the wide tile (or on a touch screen, hold the tile briefly and let go), and then click **Resize**in the app bar. You can choose between Small, Medium, and Wide. Feel free to make a large tile as well.

11. Return to Visual Studio and stop debugging.

Exercise 2: LoadingRecipe Data

The project already includes sample data, but you will want to replace it with data of your own. In Exercise 2, you will replace the sample data with real recipe data, complete with recipe images.

Task 1 – Import Recipe Data Classes

The first step is to replace the sample data classes provided by Visual Studio with recipe data classes.

1. Right-click the **DataModel** folder in Solution Explorer and use the **Add > Existing Item** command to import **RecipeDataSource.cs**and **RecipeData.json**from the data folder of the starting materials.

   Note: Visual Studio provided you with a file named SampleDataSource.cs that contains data classes named**SampleDataItem**, **SampleDataGroup**, and **SampleDataSource**. RecipeDataSource.cs contains versions of those same classes adapted to recipe data: **RecipeDataItem**, **RecipeDataGroup**, and **RecipeDataSource**. RecipeDataSource.cs contains methods named **GetGroupsAsync,** which loads recipe data from the files you just added, or from Windows Azure. It also includes all the **Windows.Data.Json** code needed to parse the JavaScript Object Notation (JSON) recipe data and load it into instances of **RecipeDataItem** and **RecipeDataGroup**. Feel free to look inside to understand how it loads and consumes the data. In particular, check out the how it usesthe Windows Runtime **HttpClient** class to load recipe data from the cloud.

2. Open **GroupedItemsPage.xaml.cs** and change all references to **SampleDataSource** class to the **RecipeDataSource** class, all references to the **SampleDataGroup** class to the **RecipeDataGroup**class, and all references to the **SampleDataItem** class to the **RecipeDataItem** class.
3. Do the same in **GroupDetailPage.xaml.cs**.
4. Do the same in **ItemDetailPage.xaml.cs**.

Task 2 – Load Recipe Pictures

The next task is to import recipe images and modify the app to load recipe data.

1. Add a folder named **Images** to the project.
2. Import the folders named tiles, 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 is important to put these folders in the Images folder, because the URLs in RecipesData.json assume that is where they are located.

> **Note:**An easy way to import the folders is to drag them from File Explorer in Windows and drop them onto the Images folder in Solution Explorer.

> **Note:**The**RecipeDataSource.GetGroups** method reads JSON recipe data from the RecipeData.json file that you imported. The image URLs in RecipeData.jsonrefer to images in the project's Images folder. If you prefer, you can download recipe data from Windows Azure by setting the **useLocalData** variable to false in **GetRecipeDataAsync**. Recipe data and images will then come from the cloud rather than from local resources. If you decide to go this route, you can remove Recipes.txt from the project. However, the Images folder must remain because it contains 150 x 150-pixel recipe images that are used to create secondary tiles in Lab 6. Secondary tile images must be local resources; they cannot be loaded remotely.

Task 3 – Test the Results

Now it is time to run the app and see how Contoso Cookbook has changed.

1. Press **F5** to debug the app and verify that the start page looks like this.

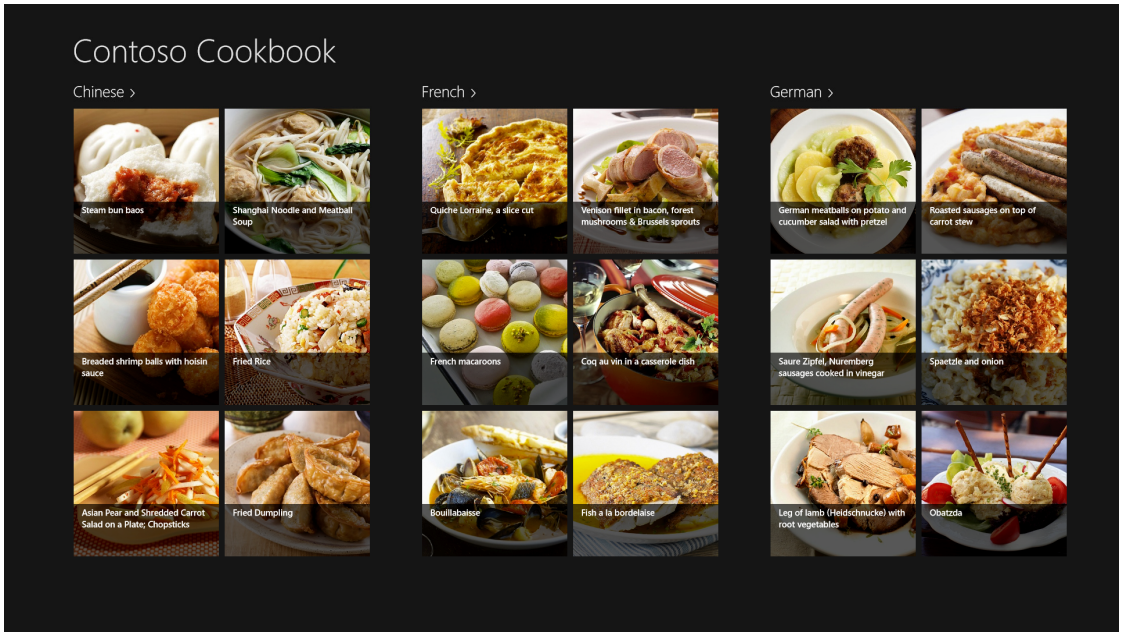

Figure 8: The Start Page with Recipes

2. Return to Visual Studio and stop debugging.

Exercise 3: Customize the UI

That is a great start, considering that we have 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 will 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

Let us begin by modifying the Start page to improve the look of the recipe items.

1. Open **GroupedItemsPage.xaml**.
2. Find the **DataTemplate** element in the GridView that has a Grid with a width and Height of 250. This is the data template used to render recipe items on the Start page.
3. Remove the final **TextBlock** element in the data template (the **TextBlock** whose **Text** property is bound to "Subtitle"), because the **RecipeDataItem** class does not have a **Subtitle** property.
4. In the same data template, change the width and height of the **Grid** element 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 overlay at the bottom of each item. When you are done, your code will look like this.

XAML

```
<GridView.ItemTemplate>
<DataTemplate>
<Grid HorizontalAlignment="Left" Width="320" Height="240">
<Border Background="{ThemeResource
ListViewItemPlaceholderBackgroundThemeBrush}">
<Image Source="{Binding ImagePath}" Stretch="UniformToFill"
AutomationProperties.Name="{Binding Title}"/>
</Border>
<StackPanel VerticalAlignment="Bottom" Background="{ThemeResource
ListViewItemOverlayBackgroundThemeBrush}">
<TextBlock Text="{Binding Title}" Foreground="{ThemeResource
ListViewItemOverlayForegroundThemeBrush}" Style="{StaticResource
TitleTextBlockStyle}" Height="48" Margin="15,0,15,0"/>
</StackPanel>
</Grid>
</DataTemplate>
</GridView.ItemTemplate>
```

5. Press **F5** to run the app. Confirm that the recipe items on the start page look like the ones shown here.

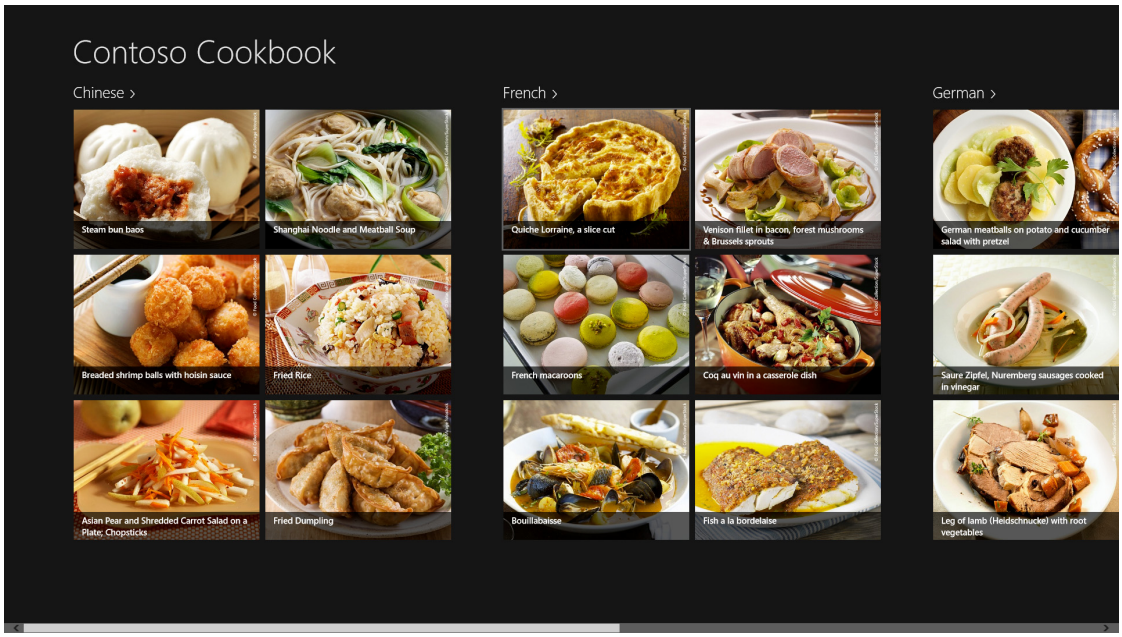

Figure 9: The New and Improved Start Page

6. Return to Visual Studio and stop debugging.

Task 2 – Modify the Group-Detail Page

You have 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 will revise that page to make group details more presentable.

1. Start the app again and tap **Chinese** in the upper-left corner of the screen to go to the group-detail page that shows Chinese recipes. The changes we will make here are minor: closing up some of the space between “Chinese” and the image below it, replacing recipe titles with short titles, and adding a preparation time to each recipe.

2. Return to Visual Studio and stop debugging.

3. Open **GroupDetailPage.xaml** and find the **GridView.Header** element. Remove the first **TextBlock**. In the Image element on the next line, replace ‘Height="400"’ with ‘Width="480"’ and change the top margin from 0 to 10. Your code should now look like this.

XAML

```
<GridView.Header>

    <StackPanel Width="480" Margin="0,4,14,0">

        <Image Source="{Binding Image}" Height="480" Margin="0,10,18
,20" Stretch="UniformToFill" AutomationProperties.Name="{Binding Tit
le}"/>

        <TextBlock Text="{Binding Description}" Margin="0,0,18,0" St
yle="{StaticResource BodyTextStyle}"/>
```

```
    </StackPanel>
</GridView.Header>
```

4. A little up in the file, find the **DataTemplate** element with a width of 480 and a height of 110. This is the data template used to render recipe items on the group-detail page.
5. Change the width of the **Grid** in the data template from 480 to 360.
6. Remove the 'Width="110"' attribute from the **Border** inside the data template to preserve the aspect ratios of the recipe images. Do leave the 'Height="110"' attribute.
7. Remove the two **TextBlock** elements whose **Text** properties are bound to the data source's **Subtitle** and **Description** properties.
8. Underneath that **TextBlock**, add the following statements to include a preparation time below the recipe title.

XAML

```
<StackPanel Orientation="Horizontal">
    <TextBlock Text="Preparation time:" Style="{StaticResource BodyTextBlockStyle}" />
    <TextBlock Text="{Binding PrepTime}" Style="{StaticResource BodyTextBlockStyle}" Margin="4,0,4,0" />
    <TextBlock Text="minutes" Style="{StaticResource BodyTextBlockStyle}" />
</StackPanel>
```

9. When you are done, here is what the modified data template should look like.

XAML

```
<DataTemplate>
<Grid Height="110" Width="360" Margin="10">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="Auto"/>
<ColumnDefinition Width="*"/>
</Grid.ColumnDefinitions>
<Border Background="{ThemeResource ListViewItemPlaceholderBackgroundThemeBrush}" Height="110">
<Image Source="{Binding ImagePath}" Stretch="UniformToFill" AutomationProperties.Name="{Binding Title}"/>
</Border>
<StackPanel Grid.Column="1" VerticalAlignment="Top" Margin="10,0,0,0">
```

```
<TextBlock Text="{Binding Title}" Style="{StaticResource
TitleTextBlockStyle}" TextWrapping="NoWrap"/>
<StackPanel Orientation="Horizontal">
<TextBlock Text="Preparation time:" Style="{StaticResource
BodyTextBlockStyle}" />
<TextBlock Text="{Binding PrepTime}" Style="{StaticResource
BodyTextBlockStyle}" Margin="4,0,4,0" />
<TextBlock Text="minutes" Style="{StaticResource BodyTextBlockStyle}" />
</StackPanel>
</StackPanel>
</Grid>
</DataTemplate>
```

10. Start the app and tap any group header. Verify that your group-detail page resembles this one.

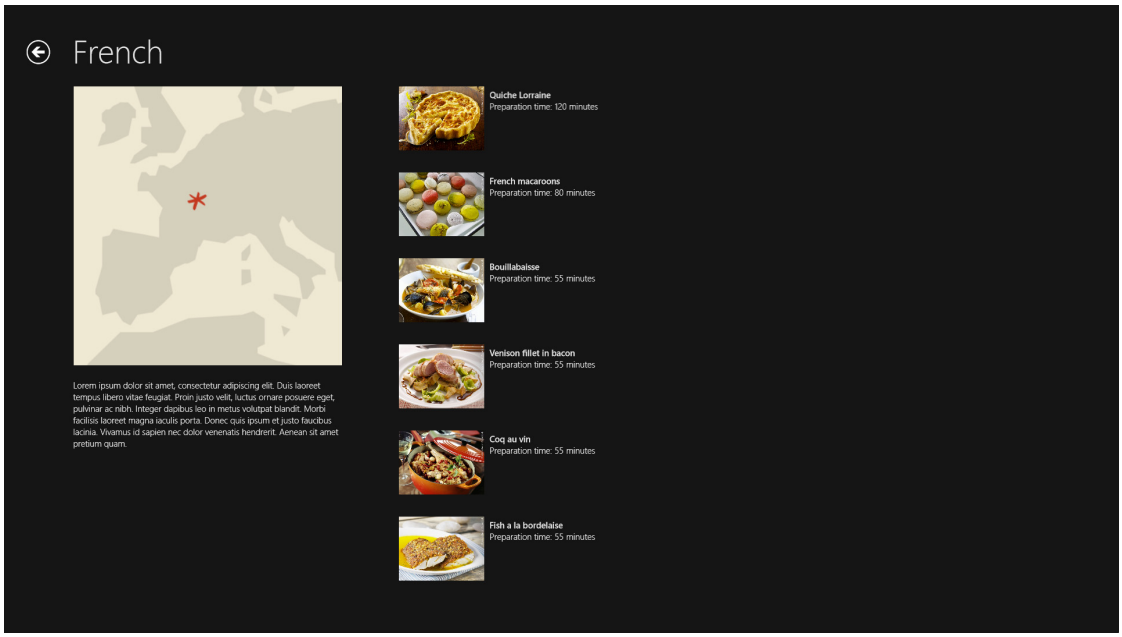

Figure 10: The Modified Group-Detail Page

11. Return to Visual Studio and stop debugging.

Task 3 – Modify the Item-Detail Page

The final task in crafting a basic UI for the app is to modify the item-detail page to present more info about recipes, including directions and ingredients.

1. Run the app and tap **Fried Dumpling**. Clearly, we have some work to do on the item-detail page.

2. Return to Visual Studio and stop debugging.
3. 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**.
   A. Hint: To add the new class, select the **Code** group under Visual C# and then select **Class**.
4. Replace the file's contents with this code.

C#

```
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Windows.UI.Xaml.Data;

namespace ContosoCookbook.Common
{
    class ListConverter : IValueConverter
    {
        public object Convert(object value, Type targetType,
object parameter, string language)
        {
            ObservableCollection<string> items = (ObservableCollecti
on<string>)value;
            StringBuilder builder = new StringBuilder();

            foreach (var item in items)
            {
                builder.Append(item);
                builder.Append("\r\n");
            }

            return builder.ToString();
        }
```

```
        public object ConvertBack(object value, Type targetType, obj
ect parameter, string language)
        {
            throw new NotImplementedException();
        }
    }
}
```

Note: ListConverter is a value converter that converts an array of strings into a single string containing line breaks. We need it because we will be binding the Text property of a **TextBlock** to an array of strings, and that requires a value converter.

5. Open **ItemDetailPage.xaml** and add the following statement to the <Page.Resources> section near the top of the file to declare a **ListConverter** instance.

XAML

```
<Page.Resources>
<common:ListConverter x:Key="ListConverter" />
    </Page.Resources>
```

6. Replace the **Grid** element with this one.

XAML

```
<Grid Grid.Row="1" x:Name="contentRegion">
<ScrollViewer x:Name="landscapeContent" Grid.Row="1">
<Grid Margin="120,0,20,20">
<Grid.ColumnDefinitions>
<ColumnDefinition Width="400" />
<ColumnDefinition Width="40" />
<ColumnDefinition Width="360" />
<ColumnDefinition Width="40" />
<ColumnDefinition />
</Grid.ColumnDefinitions>

<StackPanel Orientation="Vertical" Grid.Column="0">
<Image Width="400" Margin="0,20,0,10" Stretch="Uniform" Source="{Binding
ImagePath}"/>
<StackPanel Orientation="Horizontal">
<TextBlock FontSize="26.667" FontWeight="Light" Text="Preparation time:"/>
<TextBlock FontSize="26.667" FontWeight="Light" Text="{Binding PrepTime}"
Margin="10,0,8,0"/>
<TextBlock FontSize="26.667" FontWeight="Light" Text="minutes"/>
</StackPanel>
</StackPanel>

<StackPanel Orientation="Vertical" Grid.Column="2">
<TextBlock FontSize="26.667" FontWeight="Light" Text="Ingredients"
Margin="0,0,0,16"/>
```

```
<TextBlock FontSize="20" FontWeight="Light" LineHeight="28" Text="{Binding
Ingredients, Converter={StaticResource ListConverter}}" TextWrapping="Wrap"
/>
</StackPanel>

<StackPanel Orientation="Vertical" Grid.Column="4">
<TextBlock FontSize="26.667" FontWeight="Light" Text="Directions"
Margin="0,0,0,16"/>
<TextBlock FontSize="20" FontWeight="Light" Text="{Binding Directions}"
Margin="0,0,40,0" TextWrapping="Wrap" />
</StackPanel>
</Grid>
</ScrollViewer>

<ScrollViewer x:Name="portraitContent" Grid.Row="1" Visibility="Collapsed">
<StackPanel Orientation="Vertical" Margin="120,0,20,20">
<Image Width="400" Margin="0,20,0,10" Stretch="Uniform" Source="{Binding
ImagePath}" HorizontalAlignment="Left"/>
<StackPanel Orientation="Horizontal">
<TextBlock FontSize="26.667" FontWeight="Light" Text="Preparation time:"/>
<TextBlock FontSize="26.667" FontWeight="Light" Text="{Binding PrepTime}"
Margin="10,0,8,0"/>
<TextBlock FontSize="26.667" FontWeight="Light" Text="minutes"/>
</StackPanel>
<TextBlock FontSize="26.667" FontWeight="Light" Text="Ingredients"
Margin="0,24,0,8"/>
<TextBlock FontSize="20" FontWeight="Light" LineHeight="28" Text="{Binding
Ingredients, Converter={StaticResource ListConverter}}" TextWrapping="Wrap"
/>
<TextBlock FontSize="26.667" FontWeight="Light" Text="Directions"
Margin="0,24,0,8"/>
<TextBlock FontSize="20" FontWeight="Light" Text="{Binding Directions}"
Margin="0,0,40,0" TextWrapping="Wrap" />
</StackPanel>
</ScrollViewer>
</Grid>
```

> **Note:** The new data template shows recipes in 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.

7. Now run the app again. Tap **Fried Dumpling** and verify that the item-detail page looks like the one in Figure 11.

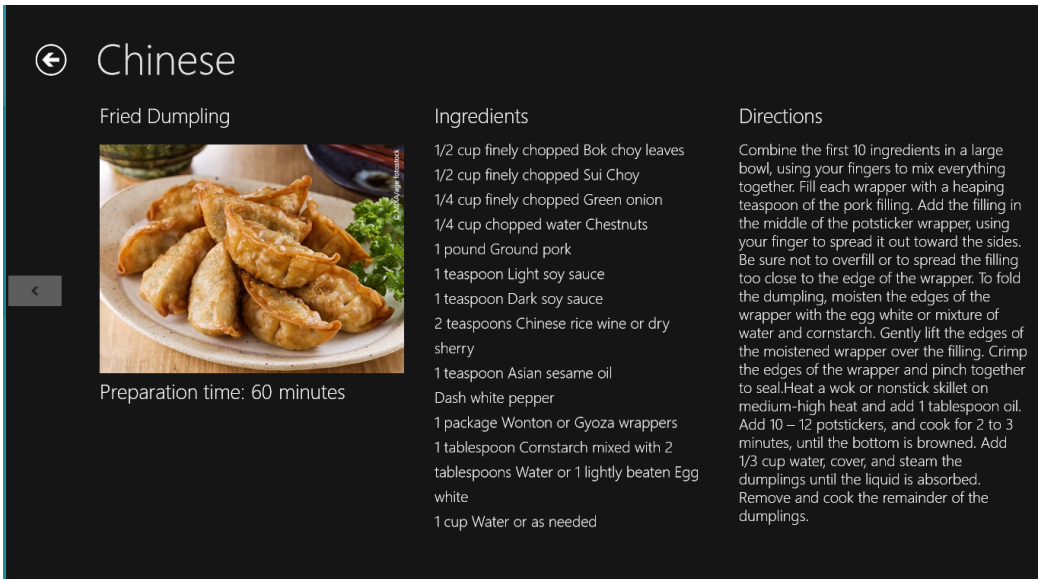

Figure 11: The Modified Item-Detail Page

8. Return to Visual Studio and stop debugging.

Summary

In this lab, you created a new Windows Store app using the Grid App project in Visual Studio. Then you replaced the sample data with real data, replaced the default branding assets with ones tailored to the app, and customized the UI by modifying 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.

You also imported code that demonstrates how the **HttpClient** class can be used to load data from a remote data source and how the **Windows.Data.Json** classes in the Windows Runtime can be used to consume JSON data in C#. By modifying data templates, you customized the way this data is presented to the user.

There is still more to do to make Contoso Cookbook a first-class app for the Windows Store. The journey continues in lab 2.