Hands-On Lab

Creating a Windows 8 Metro style app

Lab version: 1.0.0

Last updated: 6/19/2012



Contents

[Overview 3](#_Toc326940326)

[Exercise 1: Create a Metro style app 4](#_Toc326940327)

[Task 1 – Create the Project 4](#_Toc326940328)

[Task 2 – Familiarize Yourself with the Project 8](#_Toc326940329)

[Task 3 – Customize the Start Page 8](#_Toc326940330)

[Task 4 – Customize the Branding 9](#_Toc326940331)

[Exercise 2: Load Recipe Data 11](#_Toc326940332)

[Task 1 – Import Recipe Data 11](#_Toc326940333)

[Task 2 – Load Recipe Data 11](#_Toc326940334)

[Task 3 – Test the Results 12](#_Toc326940335)

[Exercise 3: Customize the UI 13](#_Toc326940336)

[Task 1 – Modify the Start Page 13](#_Toc326940337)

[Task 2 – Modify the Group-Detail Page 15](#_Toc326940338)

[Task 3 – Modify the Item-Detail Page 16](#_Toc326940339)

[Summary 20](#_Toc326940340)

Overview

* 1. Contoso Cookbook is a series of hands on labs designed to immerse you into Windows 8 Metro style app development. Through this series, you will create a beautiful, functional, real world, Metro style app that leverages some of the new key features available in Windows 8. By the end of the series, you will know how to build an application that demonstrates some of the key characteristics of a great Metro style app, including:
* A “Metro” user experience that leverages the signature Windows 8 controls such as ListView, 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 life cycle 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 HTML and Javascript to create the application, implement the navigation, download the data from a REST based service, 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 XMLHttpRequest via WinJS.xhr to retrieve recipe data from a REST service
  + Consume that data and data-bind to a ListView 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

# Setup

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

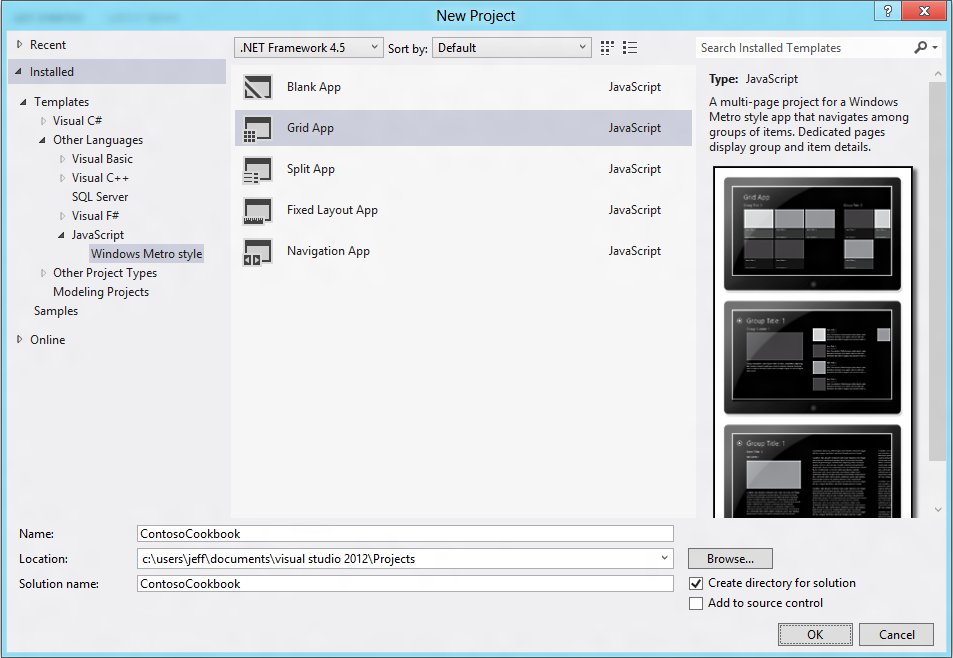
# Exercises

* 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 JavaScript 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 JavaScript project named “ContosoCookbook.” Be sure to select “Windows Metro Style” from the list of JavaScript templates, and to select “Grid App” from the list of template types, as shown in Figure 1.
     1. 
     2. Figure 1
     3. Creating the ContosoCookbook project
  3. 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*.
     1. 
     2. Figure 2
     3. The Contoso Cookbook start page
  4. 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 ListView control. If you’re curious about how it’s declared, open groupedItems.html in the pages\groupedItems folder and look for the DIV with the data-win-control=”WinJS.UI.ListView” attribute near the bottom of the page. Without this attribute, it’s an ordinary DIV. With it, however, it’s a ListView control that supports data binding and templating. Under the hood, WinJS scans the DOM, sees the data-win-control attribute, and converts the DIV into a ListView.
  5. Find out what happens if you touch or click on one of the ListView items. For example, tap the item labeled “Item Title: 1” to display the screen shown in Figure 3. This is the *item-detail page*.
     1. **Note:**  Windows 8 is 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!
     2. 
     3. Figure 3
     4. The item-detail page
  6. 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.
  7. Tap “Group Title: 1” under “ContosoCookbook” in the upper-left corner of the start page to display the *group-detail page* (Figure 4).
     1. 
     2. Figure 4
     3. The group-detail page
  8. 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.

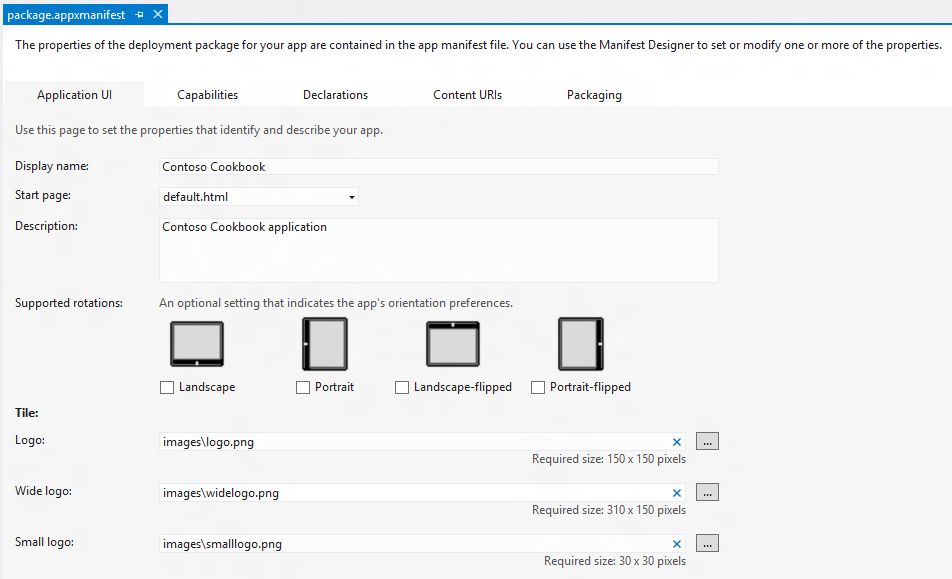
Task 2 – Familiarize Yourself with the Project

* 1. It’s clear that when Visual Studio generated the project, it gave you a lot for free. Specifically, it gave you several HTML pages plus CSS and JavaScript to go with them, logic and UI for navigating between pages, and sample data resources. To implement Contoso Cookbook, we’ll leverage what Visual Studio generated. 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 pages folder. You’ll find three subfolders there, each corresponding to one of the pages in the app:
     + groupedItems, which contains the source code files for the app’s start page
     + itemDetail, which contains the source code files for the item-detail page
     + groupDetail, which contains the source code files for the group-detail page
  3. In addition to the three pages in the pages folder, the project contains a page named default.html, which hosts the other pages. Open default.html and examine the DIV in the BODY element. It defines the content area in which the other pages are displayed.
     1. **Note:** The BODY element contains a second DIV that’s commented out. That DIV, whose ID is “appbar,” represents an application bar containing buttons, or “commands,” that correspond to common actions users can perform in the application. In subsequent labs, you will uncomment that DIV and add commands to it.
  4. Examine the contents of the project’s js folder. Here’s a quick summary of the files that are found there:
     + data.js, which contains sample data and code to bind that data to controls
     + default.js, which contains the code-behind for default.html
     + navigator.js, which contains helper functions for navigation
  5. Look in the project’s images folder, where you’ll find image assets used to brand the app.

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 groupedItems.html in Visual Studio.
  3. Find the SPAN element whose class is “pagetitle” and change “ContosoCookbook” to “Contoso Cookbook,” as shown below:
     1. HTML
     2. <span class="pagetitle">Contoso Cookbook</span>
  4. 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).
     1. 
     2. Figure 5
     3. The modified start page
  5. Return to Visual Studio and use the **Stop Debugging** command to close the application.

Task 4 – Customize the Branding

* 1. If you go out to the Windows 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 images folder. In this task, you’ll replace the logo that Visual Studio generated with one more suitable for a cookbook application. While you’re at it, you’ll replace the other PNGs in the images folder to uniquely brand the application, and finish up by modifying the application manifest.
  2. 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 application and remove the tile.
  3. Go back to Visual Studio and right-click the images folder. Then use the **Add - Existing Item** command to import Logo.png, SmallLogo.png, SplashScreen.png, StoreLogo.png, and WideLogo.png from the Images folder of the lab starting materials. When prompted, allow these files to overwrite the existing files with the same names.
  4. 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.
  5. Change the application’s display name to “Contoso Cookbook” and its description to “Contoso Cookbook application,” as shown in Figure 6. Also enter “images\widelogo.png” into the Wide Logo box to give the application a wide tile.
     1. 
     2. Figure 6
     3. Changing the branding in the manifest
  6. Press F5 to launch the application.
  7. Watch as the application starts up. Is the splash screen (the screen that’s briefly shown as the app loads) different than before?
  8. Go to the Window 8 start screen and confirm that it contains a tile like the one below.
     1. 
     2. Figure 7
     3. The new application tile
     4. **Note:** If you’d prefer a square tile, right-click the wide tile (or on a touch screen, drag the tile down slightly and let go), and then click “Smaller” in the application bar.
  9. Return to Visual Studio and stop debugging.

Exercise 2: Load Recipe Data

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

Task 1 – Import Recipe Data

* 1. The first task is to import recipe data and images.
  2. Create a new folder named “data” in the ContosoCookbook project. You can do this by right-clicking on the project in Solution Explorer and selecting **Add - New Folder**.
  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.
     1. **Note:** If you take a moment to look inside Recipes.txt, you’ll see that it contains JSON-encoded data denoting recipes and recipe groups.
  4. Copy 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 is important to put them in the images folder, because the image URLs in Recipes.txt assume that is where they are 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 onto the Images folder in Solution Explorer.

Task 2 – Load Recipe Data

* 1. The next step is to replace the sample data in data.js (which Visual Studio placed in the project’s js folder) with code that loads recipe data.
  2. Open data.js and remove (or comment out) all the lines between “use strict” and the getItemReference function.
     1. **Note:**  The lines you just removed declare and initialize arrays named sampleGroups and sampleItems containing sample data. You no longer need these arrays since you’re replacing the sample data with real data.
  3. Further down in the file, remove or comment out the following statements:
     1. JavaScript
     2. sampleItems.forEach(function (item) {
     3. list.push(item);
     4. });
  4. Replace the forEach loop that you just deleted with the following statements:
     1. JavaScript

WinJS.xhr({url: "data/Recipes.txt"}).then(function (xhr) {

* + 1. var items = JSON.parse(xhr.responseText);
    2. // Add the items to the WinJS.Binding.List
    3. items.forEach(function (item) {
    4. list.push(item);
    5. });

});

* + 1. **Note:** The code you just added uses WinJS’s WinJS.xhr function to load the Recipes.txt file you imported in the previous task. The image URLs in Recipes.txt refer to images in the project’s images folder. If you’d prefer, you can download recipe data from Azure by changing the URL passed to WinJS.xhr to “http://contosorecipes8.blob.core.windows.net/AzureRecipesRP.” 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 150x150 recipe images that will be used to create secondary tiles in Lab 6. Secondary tile images must be local resources; they cannot be loaded remotely
    2. Even though you’re using XMLHttpRequest, you don’t have to worry about cross-origin restrictions in Metro. The call succeeds even if the target lives in a different domain!

Task 3 – Test the Results

* 1. Now it’s time to run the app and see how Contoso Cookbook has changed.
  2. Press F5 to debug the application and verify that the start page looks like the one below.
     1. 
     2. Figure 8
     3. The start page with recipes
  3. Return to Visual Studio and stop debugging.

Exercise 3: Customize the UI

Did you notice that the word “undefined” appeared at the bottom of each recipe on the start page? That’s because groupedItems.html contains a data template that binds properties of our recipe objects to the ListView control that renders the recipes. An *item template* is an HTML fragment that tells a control how to render each item data-bound to it. The default item template provided by Visual Studio links the text at the bottom of the item to a property named subtitle, but the subtitle property doesn’t exist in the recipe data loaded by the application.

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 groupedItems.html. Find the DIV whose class is “itemtemplate” and remove the H6 element.
  3. In the H4 element on the line above the one you just deleted, change “textContent: title” to “textContent: shortTitle.”
  4. Now let’s modify the CSS styles that Visual Studio generated to decrease the height of the overlay at the bottom each item (the partially transparent black box in which the recipe titles appear). While we are at it, we will also modify the recipe dimensions to preserve the images’ aspect ratios. Start by opening groupedItems.css and making the changes to the -ms-grid-rows, height and width properties as demonstrated below:
     1. CSS
     2. .groupeditemspage .groupeditemslist .win-item {
     3. -ms-grid-columns: 1fr;
     4. -ms-grid-rows: 1fr 48px;
     5. display: -ms-grid;
     6. height: 240px;
     7. width: 320px;
     8. }
  5. A little further down in groupedItems.css, make a modification to the -ms-grid-rows property:
     1. CSS
     2. .groupeditemspage .groupeditemslist .win-item .item-overlay {
     3. -ms-grid-row: 2;
     4. -ms-grid-rows: 1fr 1px;
     5. display: -ms-grid;
     6. padding: 6px 15px 2px 15px;
     7. }
  6. Now press F5 to run the application. Confirm that the recipe items on the start page look like the ones below.
     1. 
     2. Figure 9
     3. The new and improved start page
  7. Return to Visual Studio and stop debugging.

Task 2 – Modify the Group-Detail Page

* 1. 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.
  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. Observe that the word “undefined” appears above the Chinese logo image and in the recipe tiles. Just like before, the default template is looking for properties that are not in our data model.
  3. Return to Visual Studio and stop debugging.
  4. Open groupDetail.html. Find the DIV whose class is “headerTemplate,” and remove the H2 element whose class is “group-subtitle.”
  5. Now find the item template – the DIV whose class is “itemtemplate” - that defines the look of the recipe thumbnails on the right half of the page – and remove the H6 element that is bound to the subtitle; also remove the H4 element (bound to the description) that follows the subtitle element we just removed. Change “textContent: title” to “textContent: shortTitle” in the remaining H4 element so that the item template looks like this:
     1. HTML
     2. <div class="itemtemplate" data-win-control="WinJS.Binding.Template">
     3. <img class="item-image" src="#" data-win-bind="src: backgroundImage; alt: title" />
     4. <div class="item-info">
     5. <h4 class="item-title" data-win-bind="textContent: shortTitle"></h4>

</div>

</div>

* 1. Now add the following statements in place of the ones you just deleted, after the H4 element and before the closing DIV tag:
     1. HTML
     2. <h4 class="item-subtitle">

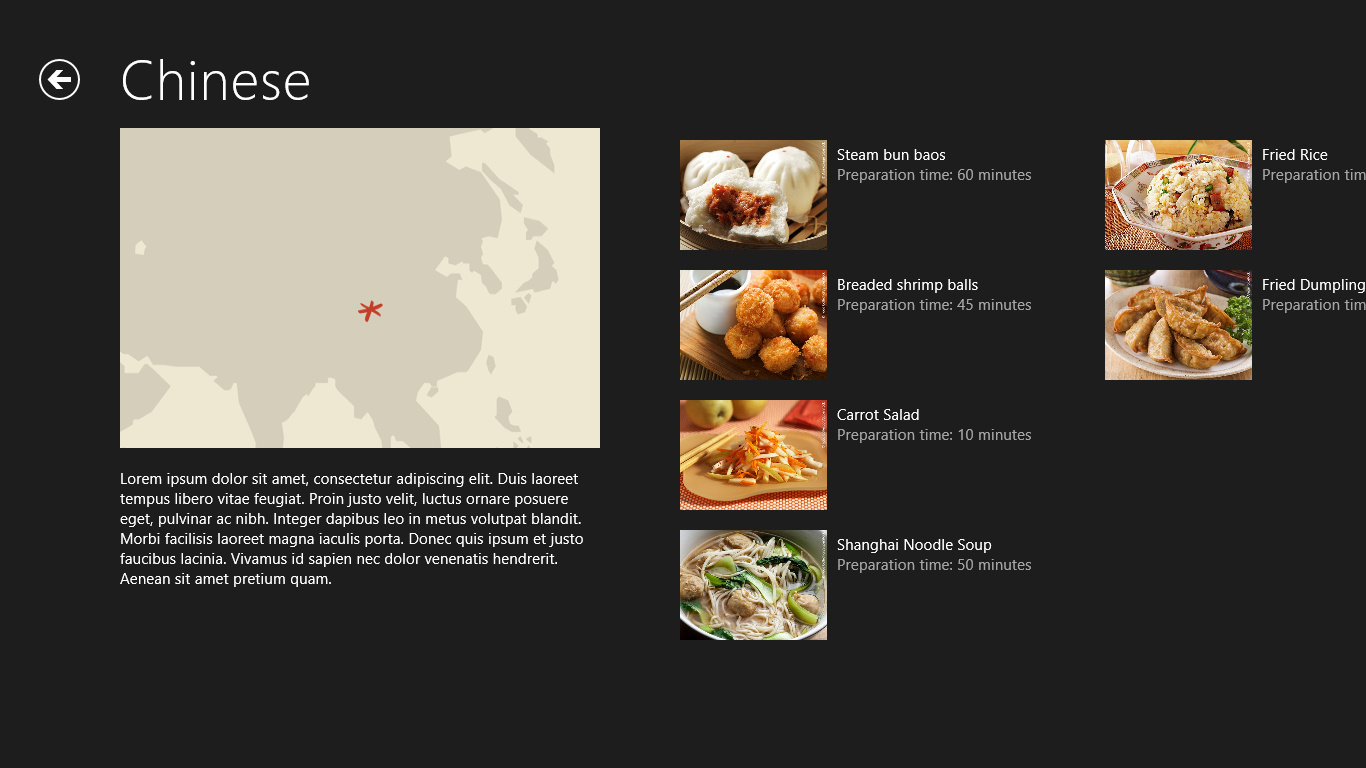
Preparation time: <span data-win-bind="textContent: preptime"></span> minutes

* + 1. </h4>
    2. **Note:** “preptime” is the name of one of the properties of the recipe objects the app loads. You are using it to include information about preparation time in each of the page’s recipe thumbnails. Note the data-win-bind attribute used to bind the SPAN element’s textContent property to the preptime property of the recipe object.
  1. Open groupDetail.css and change the height in the CSS class below to 320px to preserve the aspect ratio of the group image displayed on the left side of the page:
     1. CSS

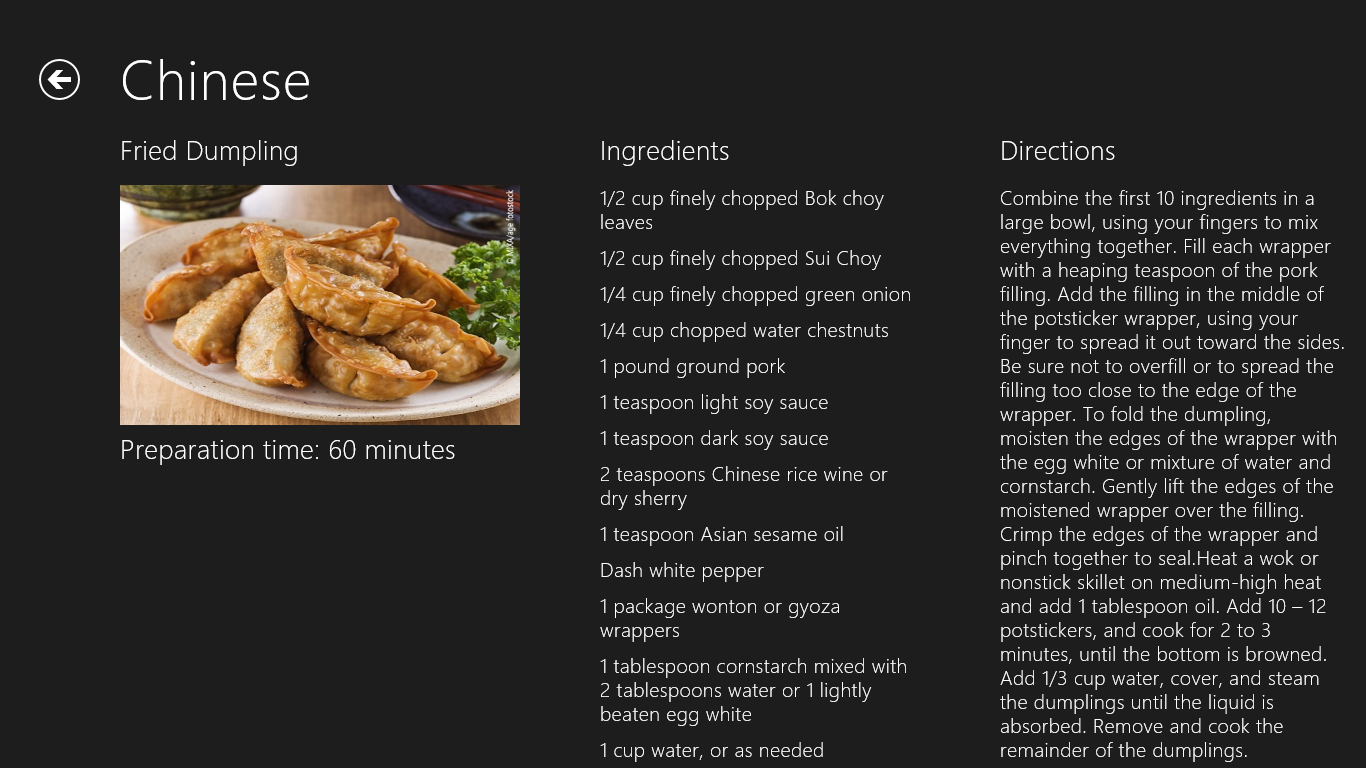
.groupdetailpage .itemslist .win-groupheader .group-image {

* + 1. -ms-grid-row: 2;
    2. background-color: rgba(147, 149, 152, 1);
    3. height: 320px;
    4. margin: 0;
    5. margin-bottom: 20px;
    6. width: 480px;

}

* 1. Also in groupDetail.css, modify the -ms-grid-columns and width properties in the “.groupdetailpage .grouplist .win-item” class as shown below. This will space the recipe items a little closer together and allow the recipe images to retain their original aspect ratios.
     1. CSS
     2. .groupdetailpage .itemslist .win-item {
     3. -ms-grid-columns: 147px 1fr;
     4. -ms-grid-rows: 1fr;
     5. display: -ms-grid;
     6. height: 110px;
     7. width: 360px;
     8. }
  2. Start the application and tap “Chinese” again. Verify that your group-detail page resembles the one below:
     1. 
     2. Figure 10
     3. The modified group-detail page
  3. 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 one of the recipe items. Clearly, we have some work to do on the item-detail page.
  3. Return to Visual Studio and stop debugging.
  4. Open itemDetail.html. Find the DIV element decorated with a role=”main” attribute and modify it to look like this:
     1. HTML
     2. <div class="content" aria-label="Main content" role="main">
     3. <article>
     4. <div>
     5. <header>
     6. <h2 class="item-title"></h2>
     7. </header>
     8. <img class="item-image" src="#" />
     9. <h2>Preparation time: <span class="item-subtitle"></span> minutes
     10. </h2>
     11. </div>
     12. <div class="ingredients">
     13. <h2>Ingredients</h2>
     14. <div class="item-ingredients"></div>
     15. </div>
     16. <div class="directions">
     17. <h2>Directions</h2>
     18. <h2 class="item-directions"></h2>
     19. </div>
     20. </article>
     21. </div>
  5. The item-detail page doesn’t use declarative data binding; it simply assigns properties of a recipe object to HTML elements in the code-behind. To that end, open itemDetail.js and replace the ready function with the one shown here:
     1. JavaScript
     2. ready: function (element, options) {
     3. var item = options && options.item ? Data.resolveItemReference(options.item) : Data.items.getAt(0);
     4. element.querySelector(".titlearea .pagetitle").textContent = item.group.title;
     5. element.querySelector("article .item-title").textContent = item.title;
     6. element.querySelector("article .item-subtitle").textContent = item.preptime;
     7. element.querySelector("article .item-image").src = item.backgroundImage;
     8. element.querySelector("article .item-image").alt = item.shortTitle;
     9. // Display ingredients list
     10. var ingredients = element.querySelector("article .item-ingredients");
     11. for (var i = 0; i < item.ingredients.length; i++) {
     12. var ingredient = document.createElement("h2");
     13. ingredient.textContent = item.ingredients[i];
     14. ingredient.className = "ingredient";
     15. ingredients.appendChild(ingredient);
     16. }
     17. // Display cooking directions
     18. element.querySelector("article .item-directions").textContent = item.directions;
     19. element.querySelector(".content").focus();
     20. }
  6. To complete the modifications, you also need to modify the page’s CSS. Open itemDetail.css and add the following statements after the “.itemdetailpage .content article header .item-subtitle” class:
     1. CSS
     2. .itemdetailpage .content article .ingredients {
     3. -ms-grid-column: 3;
     4. margin-left: 40px;
     5. }
     6. .itemdetailpage .content article .ingredients .item-ingredients {
     7. margin-top: 20px;
     8. }
     9. .itemdetailpage .content article .ingredients .item-ingredients .ingredient {
     10. padding-bottom: 12px;
     11. font-size: 20px;
     12. }
     13. .itemdetailpage .content article .directions {
     14. -ms-grid-column: 5;
     15. margin-left: 40px;
     16. }
     17. .itemdetailpage .content article .directions .item-directions {
     18. margin-top: 20px;
     19. font-size: 20px;
     20. }
  7. Before you close itemDetail.css, find the first “.itemdetailpage .content article” class – the one near the top of the file – and replace it with the one below:
     1. CSS
     2. .itemdetailpage .content article {
     3. /\* Define a multi-column, horizontally scrolling article by default. \*/
     4. column-fill: auto;
     5. column-gap: 80px;
     6. column-width: 480px;
     7. height: calc(100% - 50px);
     8. margin-left: 120px;
     9. margin-top: 5px;
     10. margin-right: 20px;
     11. display: -ms-grid;
     12. -ms-grid-columns: 400px 40px 360px 40px 1fr;
     13. }
     14. **Note:** This modification removes the fixed width, allowing recipe data to expand horizontally to fill the page. It also declares a 5-column grid for laying out DOM elements. Grid layout is described in the CSS3 Grid Layout specification at <http://dev.w3.org/csswg/css3-grid-align/>. If you examine the CSS you added in step 5, you’ll see that we used -ms-grid-column to position the DIV containing recipe ingredients in column 3 of the grid and recipe directions in column 5. In other words, we’re using grid layout to divide the screen into columns and position HTML content in those columns.
  8. Finally, find the “.itemdetailpage .content article .item-image” class and remove the height property so that it looks like this:
     1. CSS
     2. .itemdetailpage .content article .item-image {
     3. margin-bottom: 3px;
     4. width: 400px;
     5. }
  9. Now run the application again. Tap “Fried Dumpling” and verify that the item-detail page looks like the one in Figure 11.
     1. 
     2. Figure 11
     3. The modified item-detail page
  10. Return to Visual Studio and stop debugging.

Summary

* 1. In this lab, you created a new Metro style Grid App project 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 HTML, CSS, and JavaScript 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 learned how WinJS.xhr can be used to load data from a local or remote data source, how the data can be bound to a ListView control to render data items to the screen, and how data templates are used to specify how data items are rendered into HTML. By modifying the data templates and the corresponding CSS, you customized the way recipes and recipe groups are 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!