Hands-On Lab

Orientation, Snapping, and Semantic Zoom

Lab version: 1.0.0

Last updated: 6/19/2012



Contents

[Overview 3](#_Toc327911508)

[Exercise 1: Device Orientation 4](#_Toc327911509)

[Task 1 – Test the Start Page 4](#_Toc327911510)

[Task 2 – Customize the Group-Detail Page 6](#_Toc327911511)

[Task 3 – Customize the Item-Detail Page 12](#_Toc327911512)

[Exercise 2: Snapping 16](#_Toc327911513)

[Task 1 – Run Contoso Cookbook in Snapped Mode 17](#_Toc327911514)

[Task 2 – Modify the Snapped Item-Detail Page 18](#_Toc327911515)

[Exercise 3: Semantic Zoom 20](#_Toc327911516)

[Task 1 – Add a SemanticZoom Control to the Start Page 20](#_Toc327911517)

[Task 2 – Modify groupedItems.js 21](#_Toc327911518)

[Task 3 – Modify groupedItems.css 23](#_Toc327911519)

[Summary 25](#_Toc327911520)

Overview

* 1. One of the hallmarks of a great Metro style app is that it adapts to screens of various sizes, and it handles rotations between landscape mode and portrait mode. It must also respond to *snapping*, which allows two apps to share the screen and run side by side.
  2. Another pertinent topic related to screen size is *semantic zoom*. Unlike optical zoom, which simply scales content on the screen in response to user stimuli such as pinch gestures, semantic zoom changes the representation of the content to show more or less detail as the user zooms in and out. Semantic zoom lets users navigate long lists of content without excessive amounts of scrolling; users can zoom out, find what they want, and then select it to go directly to the corresponding location.
  3. In this lab, you will build upon Lab 1 by adding three important UI-related features to Contoso Cookbook. First, you will customize the layout of the item-detail and group-detail pages when the screen is rotated. Next, you will customize the layout of the item-detail page when the application is snapped. Finally, you will implement semantic zoom in the start page, enabling users to zoom out and see all the recipe groups on a single screen.

# Objectives

* 1. This lab will show you how to:
  + Customize the UI when the device is rotated
  + Customize the UI when your application is snapped
  + Implement semantic zoom

# System Requirements

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

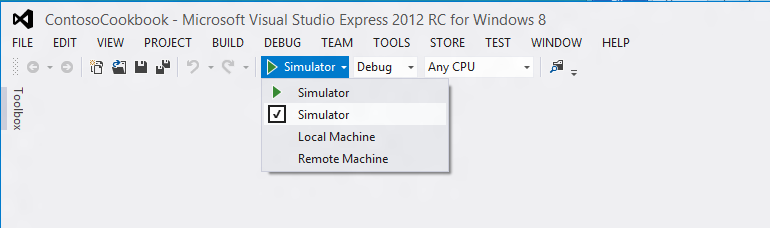
# Setup

* 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. Device Orientation
  3. Snapping
  4. Semantic Zoom
  5. Estimated time to complete this lab: **40 to 60 minutes**.

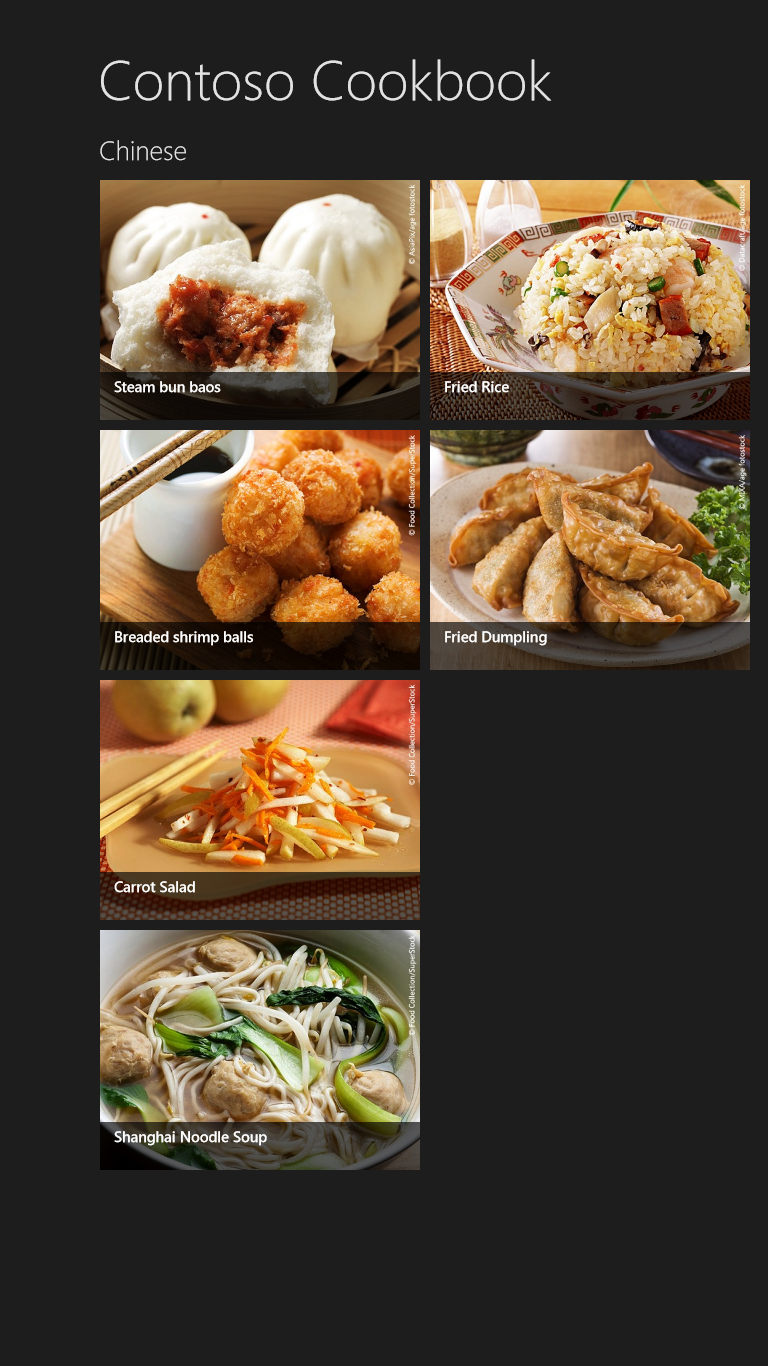
Exercise 1: Device Orientation

1. Windows 8 is designed to run on a variety of devices, including tablets and other devices that -through the aid of on-board sensors- know whether they are in landscape mode or portrait mode. A page that looks great in landscape mode may need tweaking to look equally great in portrait mode, and vice versa. In this exercise, you will modify the Contoso Cookbook app to adapt to device orientation.
   1. **Note:** You do not have to have a tablet to perform Exercise 1. You can test orientation code in the Windows Simulator, which you can access directly from Visual Studio. To target the simulator, select “Simulator” from the drop-down list below. The next time you launch the application, it will launch in the Windows Simulator.
   2. 

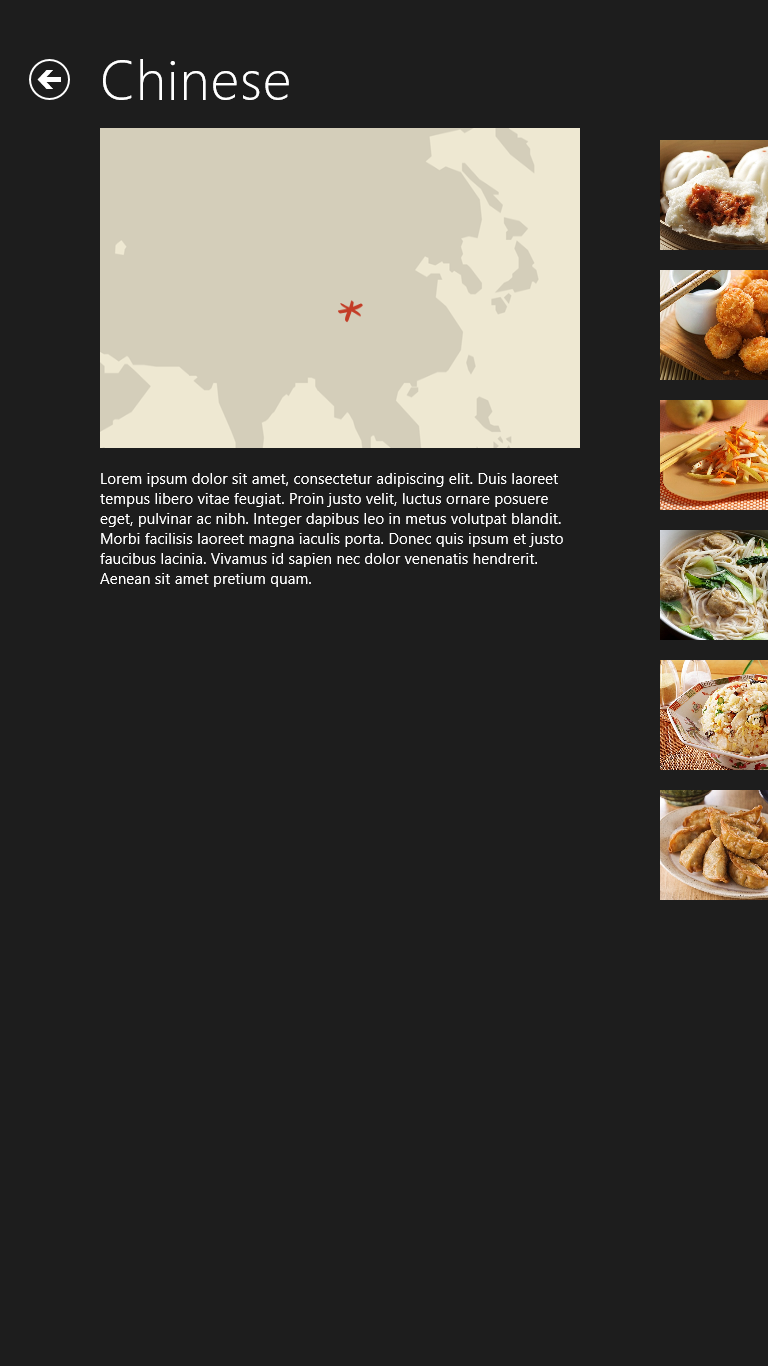
Task 1 – Test the Start Page

* 1. Let’s begin by examining the application’s start page in landscape mode and portrait mode to determine whether any changes are needed.
  2. Open the ContosoCookbook project you created in Lab 1 in Visual Studio. If you did not complete Lab 1 or would like to start with a reference copy, you will find a completed version of the lab in the starting materials.
  3. With the device in landscape mode, Press F5 to run the application in the debugger. You should see the recipes start page shown in Figure 1.
     1. 
     2. Figure 1
     3. The start page in landscape mode
  4. Now rotate the device to portrait mode. Make sure the screen rotates 90 degrees, and that the start page rotates as well and assumes the layout shown in Figure 2.
     1. **Note:** If you are testing on a tablet and the start page does not rotate, it might be because autorotation is disabled. Some devices, such as the Samsung tablets given to attendees of the September 2011 BUILD conference, have a hardware switch that locks the screen into its current orientation. If you have such a device, make sure that switch is set to autorotation mode. You can also press Win-O to toggle autorotation on and off.

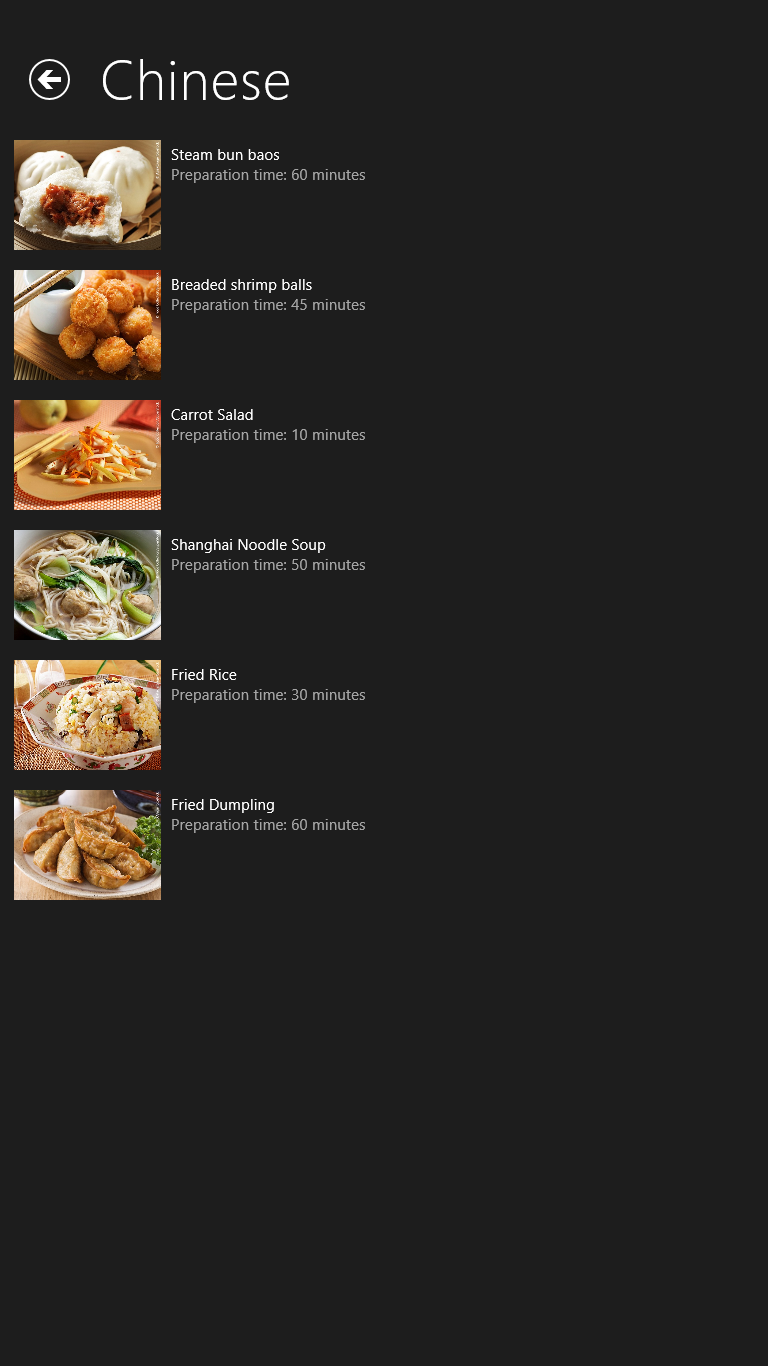
If the hardware you are testing on does not support orientation changes, remember that you can use the Windows Simulator instead. To rotate the simulator, click either of the Rotate buttons in the simulator’s right margin.

* + 1. 
    2. Figure 2
    3. The start page in portrait mode
  1. Thanks to the ListView control that renders the recipe items, the start page looks fine in both landscape and portrait modes, so no additional work is required here.
  2. Return to the Visual Studio and stop debugging.

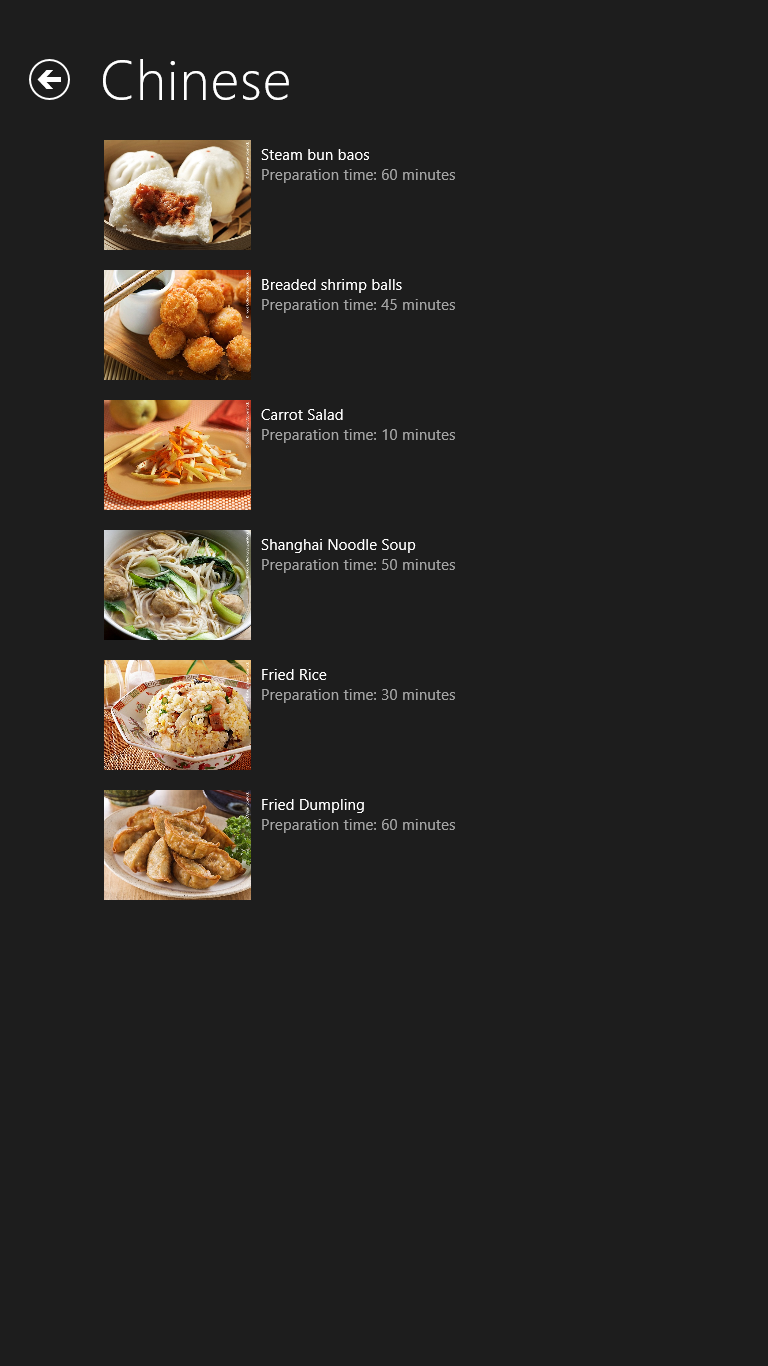
Task 2 – Customize the Group-Detail Page

* 1. The Contoso Cookbook application comprises three pages: the start page, the group-detail page, and the item-detail page. The start page does not require any modification for portrait mode, so let’s move on to the group-detail page – the one that appears when you tap a group title such as “Chinese” or “Italian.”
  2. Start the application again and tap “Chinese” to show the Chinese group-detail page. In landscape mode, the page assumes the layout in Figure 3.
     1. 
     2. Figure 3
     3. The group-detail page in landscape mode
  3. Now rotate the device to portrait mode (see Figure 4). Observe that the ListView control that renders most of the content on this page positions the group details (the recipes) to the right of the group header (the group title, image, and description), which leaves a lot of blank space in the lower half of the screen.
     1. 
     2. Figure 4
     3. The group-detail page in portrait mode
  4. Return to Visual Studio and stop debugging.
  5. Let’s reconfigure the ListView control so that when the screen is in portrait mode, recipes appear in a single vertically scrolling column underneath the group name. To begin, open groupDetail.js and find the updateLayout function. Then modify the if statement that checks for snapped view state to include a test for full-screen portrait mode as well:
     1. JavaScript
     2. // This function updates the page layout in response to viewState changes.
     3. updateLayout: function (element, viewState, lastViewState) {
     4. /// <param name="element" domElement="true" />
     5. /// <param name="viewState" value="Windows.UI.ViewManagement.ApplicationViewState" />
     6. /// <param name="lastViewState" value="Windows.UI.ViewManagement.ApplicationViewState" />
     7. var listView = element.querySelector(".itemslist").winControl;
     8. if (lastViewState !== viewState) {
     9. if (lastViewState === appViewState.snapped || viewState === appViewState.snapped ||
     10. lastViewState === appViewState.fullScreenPortrait || viewState == appViewState.fullScreenPortrait) {
     11. var handler = function (e) {
     12. listView.removeEventListener("contentanimating", handler, false);
     13. e.preventDefault();
     14. }
     15. listView.addEventListener("contentanimating", handler, false);
     16. var firstVisible = listView.indexOfFirstVisible;
     17. this.initializeLayout(listView, viewState);
     18. listView.indexOfFirstVisible = firstVisible;
     19. }
     20. }
     21. }

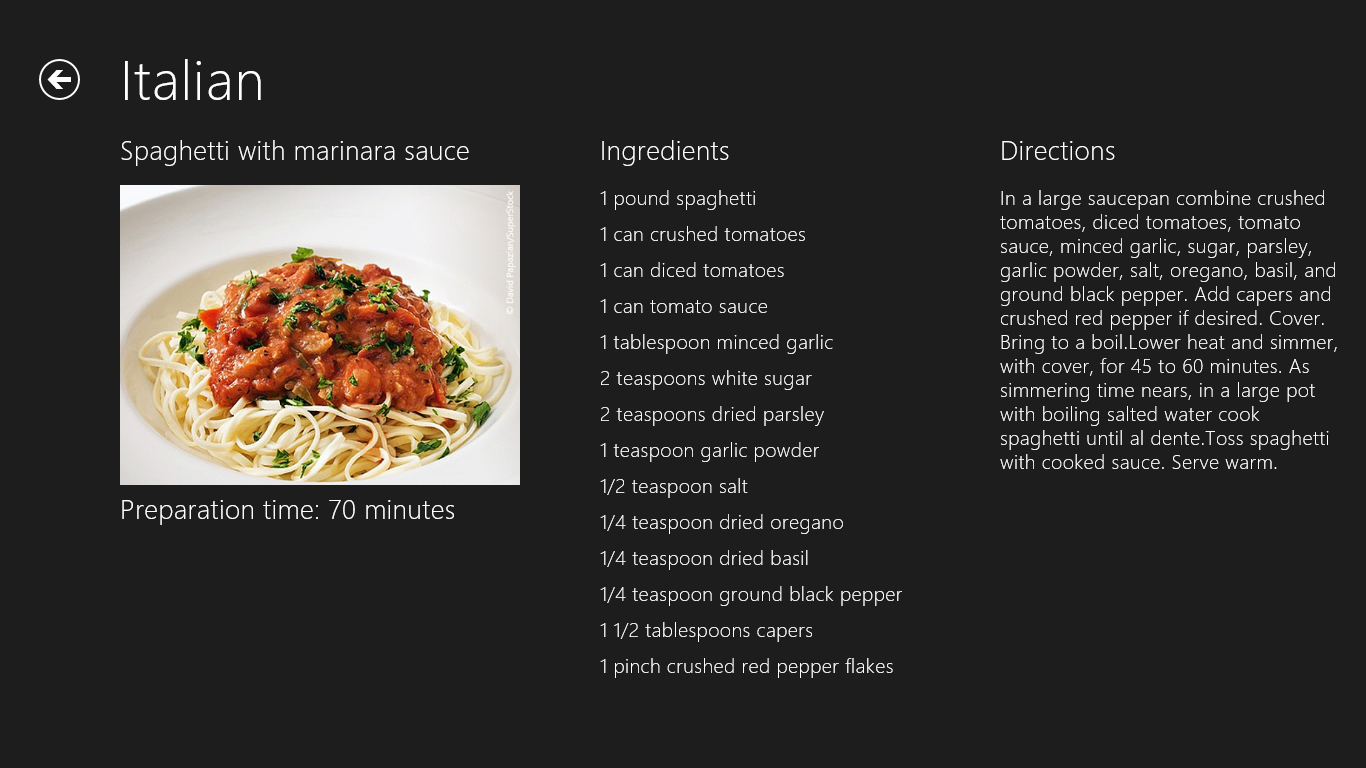
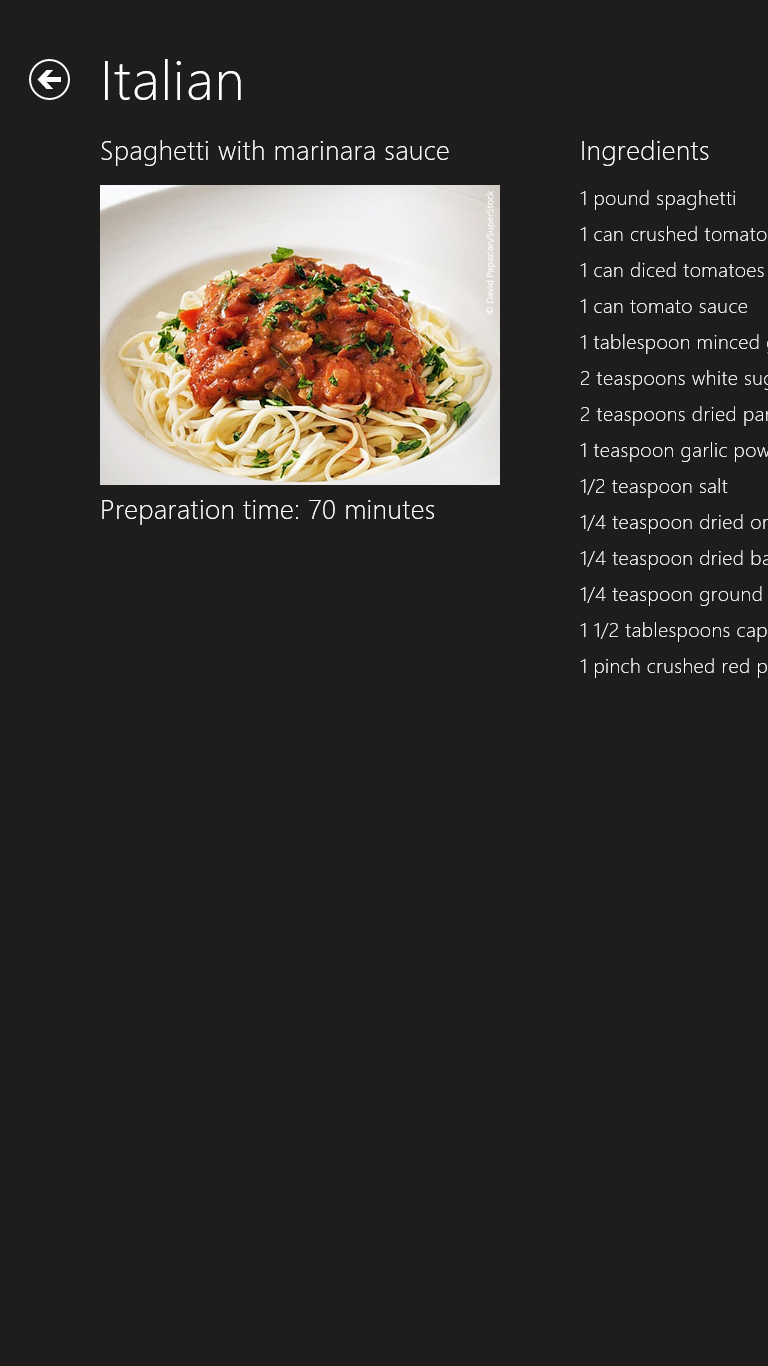
**Note:** As the comment indicates, updateLayout is called when the view state changes.   
(Metro view state has nothing to do with ASP.NET view state, so do not let the term conjure up negative images in your mind!) One of the events that precipitates a view-state change is rotating the device from landscape mode to portrait or portrait to landscape. updateLayout is also called when the application is “snapped.” Snapping is covered in the next exercise.

* 1. Next, tweak the initializeLayout function that gets called from updateLayout so full-screen portrait mode uses list layout rather than grid layout:
     1. JavaScript
     2. // This function updates the ListView with new layouts
     3. initializeLayout: function (listView, viewState) {
     4. /// <param name="listView" value="WinJS.UI.ListView.prototype" />
     5. if (viewState === appViewState.snapped || viewState === appViewState.fullScreenPortrait) {
     6. listView.layout = new ui.ListLayout();
     7. } else {
     8. listView.layout = new ui.GridLayout({ groupHeaderPosition: "left" });
     9. }
     10. },
     11. **Note:** ListView controls support two layout modes: GridLayout and ListLayout. In GridLayout mode (the default), group details appear in a grid that can be scrolled horizontally. In ListLayout mode, group details appear in a column that scrolls vertically, which is ideal for portrait-oriented displays. The statements you added to the updateLayout function change the ListView’s layout mode to ListLayout when the screen changes to portrait mode.
  2. Run the application, tap “Chinese” to display the group-detail page, and rotate the device to portrait mode. Confirm that the page assumes the layout in Figure 5.
     1. 
     2. Figure 5
     3. The modified portrait-mode layout
  3. The change to updateLayout and initializeLayout got us halfway there, but we need to increase the margin on the left side of the portrait page so that it is consistent with the margin on the landscape page. To begin, return to Visual Studio and stop debugging.
  4. Open groupDetail.css.
  5. Find the following clause near the bottom of the file:
     1. CSS
     2. @media screen and (-ms-view-state: fullscreen-portrait) {
     3. .groupdetailpage . itemslist.win-horizontal.win-viewport .win-surface {
     4. margin-left: 100px;
     5. }
     6. }
     7. **Note:** This clause is an example of a CSS3 *media query*. Media queries provide a declarative means for changing the layout of an HTML page in response to changes in width and height. Media queries are the basis for *responsive design*, a discipline in which page layouts adapt to changing dimensions without the aid of JavaScript. The CSS files generated for the Contoso Cookbook pages include CSS styles that are common to all layouts, styles that apply only when the screen is snapped, and styles that apply only in portrait mode. The CSS above corresponds to portrait mode.

Note that -ms-view-state is not part of CSS3, but is a Microsoft extension for Metro style apps.

* 1. Modify the media query by adding a second style to it:
     1. CSS
     2. @media screen and (-ms-view-state: fullscreen-portrait) {
     3. .groupdetailpage .itemslist .win-horizontal.win-viewport .win-surface {
     4. margin-left: 100px;
     5. }
     6. .groupdetailpage .itemslist .win-vertical.win-viewport .win-surface {
     7. margin-left: 90px;
     8. }
     9. }
  2. Run the application, tap “Chinese” to display the group-detail page, and rotate the device to portrait mode. Confirm that recipe items are now left-indented by 90 pixels, as shown in Figure 6
     1. 
     2. Figure 6
     3. The finished portrait-mode layout
  3. Return to Visual Studio and stop debugging.

Task 3 – Customize the Item-Detail Page

* 1. The next task is to tweak the item-detail page so that it is equally efficient in its use of space and pleasing to the eye in both landscape and portrait mode.
  2. Start the application again and tap a recipe to show the item-detail page. Figure 7 shows how the page looks in landscape mode.
     1. 
     2. Figure 7
     3. The item-detail page in landscape mode
  3. Now rotate the device to portrait mode (Figure 8). Once more, the layout that works well in landscape mode needs a bit of tweaking for portrait mode.
     1. 
     2. Figure 8
     3. The item-detail page in portrait mode
  4. Return to Visual Studio and stop debugging.
  5. The three content areas in the item-detail page – the recipe name, image, and description; the ingredients, and the directions – are DIVs in itemDetail.html. In Lab 1, you modified the CSS for those DIVs to achieve the desired layout in landscape mode. Now you will add CSS defining the layout in portrait mode. Begin by opening itemDetail.css and finding the media query below.
     1. CSS
     2. @media screen and (-ms-view-state: fullscreen-portrait) {
     3. .itemdetailpage .content article {
     4. margin-left: 100px;
     5. }

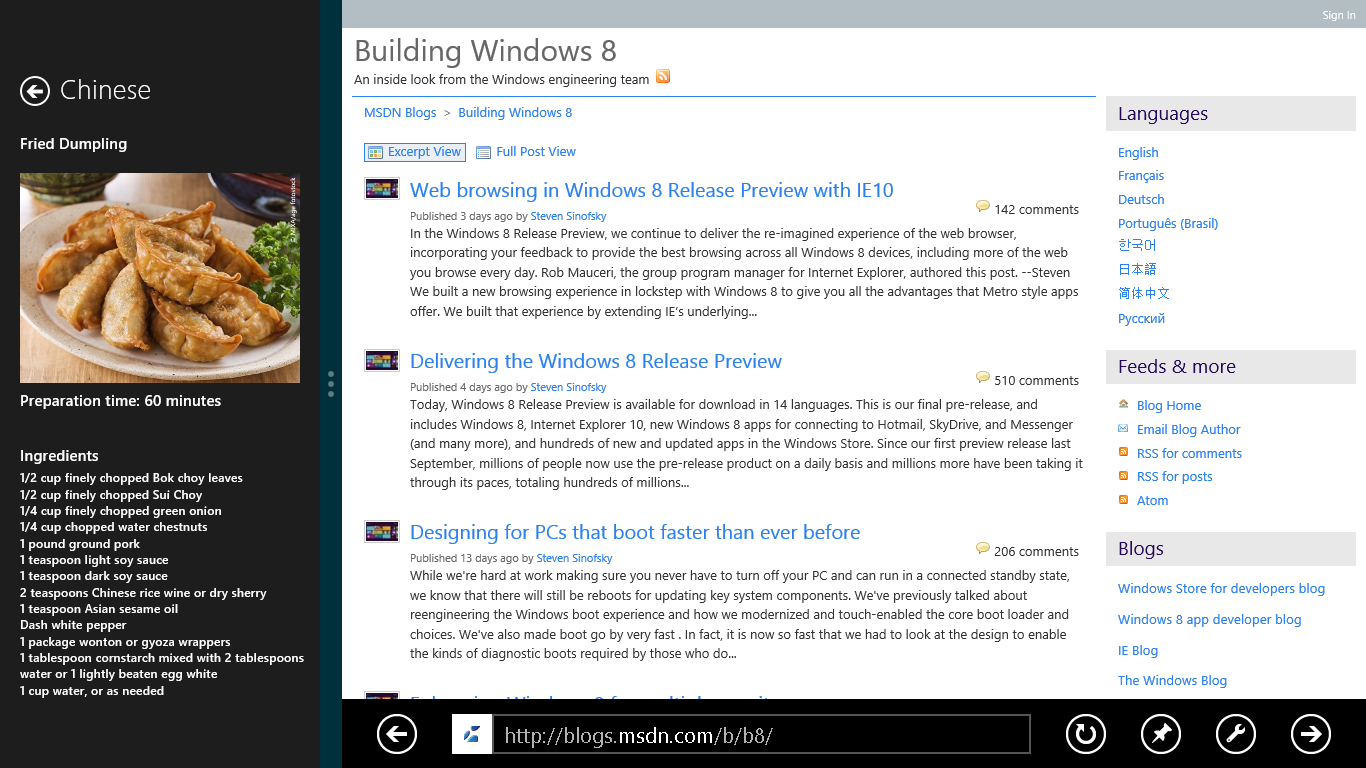
}

* + 1. Modify the media query as follows:
    2. CSS
    3. @media screen and (-ms-view-state: fullscreen-portrait) {
    4. .itemdetailpage .content article {
    5. margin-left: 100px;
    6. }

.itemdetailpage .content article .ingredients {

* + 1. -ms-grid-row: 2;
    2. -ms-grid-column: 1;
    3. margin-left: 0px;
    4. margin-top: 40px;
    5. }
    6. .itemdetailpage .content article .directions {
    7. -ms-grid-row: 3;
    8. -ms-grid-column: 1;
    9. margin-left: 0px;
    10. margin-top: 40px;
    11. }
    12. }
    13. **Note:** The default styles for .ingredients and .directions place ingredients in different columns of the layout grid. The changes you just applied keep everything in the first column, but put ingredients in the second row and directions in the third. These changes only apply to portrait mode, of course, so landscape mode should be unaffected.
  1. Start the application and tap a recipe to show the item-detail page. Rotate the display to portrait mode and verify that it assumes the single-column layout depicted in Figure 9.
     1. 
     2. Figure 9
     3. The modified portrait-mode layout
  2. Rotate back to landscape mode and make sure the page reverts to a 3-column layout.
  3. Return to Visual Studio and stop debugging.

Exercise 2: Snapping

1. Snapping enables Windows 8 users to run two Metro style apps side by side by splitting the screen horizontally. On a touch screen, you can demonstrate snapping by dragging your finger slowly across the screen starting from left edge and pausing momentarily until a snap bar – a vertical bar splitting the screen – appears. (If you do not have a touch screen, press the Windows key and the period to snap an application.) Figure 10 shows snapping in action. Contoso Cookbook occupies the left 320 pixels of the screen, while Internet Explorer occupies the remaining portion. In this context, Contoso Cookbook is the “snapped app” and Internet Explorer is the “main app” or “filled app.” If desired, the user can drag the snap bar across the screen and reverse the apps’ roles.
   1. 
   2. Figure 10
   3. Snapping in action

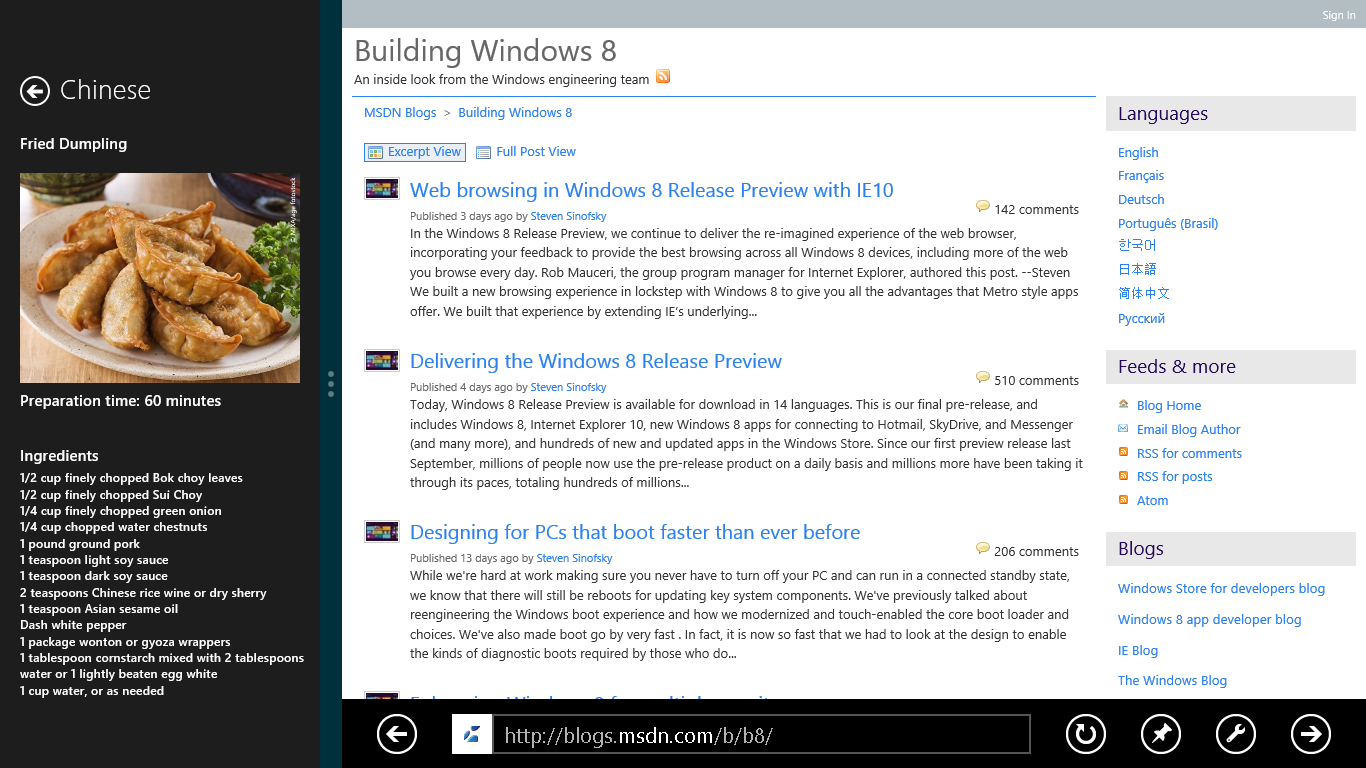
Contoso Cookbook already has some snapping behavior built in, thanks to media queries included by Visual Studio in the CSS it generated for the project. In this exercise, you will apply a simple change to improve the user experience.

* 1. **Note:** To see snapping in action, you must be running Metro on a device with a screen resolution of at least 1366 by 768 pixels. The Windows team chose 1366 as the minimum because it affords the snapped app a section of the screen that is 320 pixels wide (the same width as many smartphones) and the main app a section that is 1024 pixels wide. The snap bar consumes the extra pixels. If you are running on a lower resolution screen, use the Windows Simulator for this exercise and select a simulated screen resolution of at least 1366x768.

Task 1 – Run Contoso Cookbook in Snapped Mode

* 1. In this task, you will view each of Contoso Cookbook’s three pages when the application is snapped.
  2. Press F5 to start the application from Visual Studio. Then go back to the Windows start screen and launch another application such as Internet Explorer. With that application now occupying the screen, place a finger on the left edge of the screen and slowly drag it to the right. When Contoso Cookbook appears under your finger, pause until the snap bar appears. Then lift your finger to snap Contoso Cookbook into place.
  3. When Contoso Cookbook only occupies a portion of the screen, how does its start page differ from when it is displayed full-screen?
  4. Tap one of the group names on the start page to see the group-detail page in snapped mode. How does it differ from its full-screen equivalent?
  5. Tap one of the recipes to see the item-detail page in snapped mode. How does its content and layout compare to when it is running full screen?
  6. Return to Visual Studio and stop debugging.

Task 2 – Modify the Snapped Item-Detail Page

* 1. The default snapped layouts provided by Visual Studio were a great start, but you can customize them further. In Contoso Cookbook, it makes sense to modify the snapped item-detail page, because the default layout leaves a lot of empty real estate in the portion of the screen that the page occupies – real estate that could be used to show additional information about the recipe. Let’s modify the snapped item-detail page to include recipe ingredients.
  2. Open itemDetail.css and find the media query that reads “@media screen and (-ms-view-state: snapped).”
     1. **Note:** Just as media queries targeting -ms-view-state: fullscreen-portrait allow you to change the layout of a page when it rotates to portrait mode, media queries targeting -ms-view-state: snapped let you change the layout when the page is snapped.
  3. Replace the media query with the one below. Most of these changes are simply new styles that apply to the ingredients section of the page when snapped. The ingredients were there before; you just could not see them because they were positioned too far to the right. The new styles position the ingredients beneath the recipe image by moving them to the second grid row:
     1. CSS
     2. @media screen and (-ms-view-state: snapped) {
     3. .itemdetailpage .content {
     4. overflow-x: hidden;
     5. overflow-y: auto;
     6. }
     7. .itemdetailpage .content article {
     8. -ms-grid-columns: 300px 1fr;
     9. -ms-grid-row: 2;
     10. -ms-grid-rows: 300px auto;
     11. display: -ms-grid;
     12. height: 100%;
     13. margin-left: 20px;
     14. overflow-x: hidden;
     15. overflow-y: auto;
     16. width: 300px;
     17. }
     18. .itemdetailpage .content article .item-image {
     19. width: 280px;
     20. }
     21. .itemdetailpage .content article .item-content {
     22. padding-bottom: 60px;
     23. }
     24. .itemdetailpage .content article .ingredients {
     25. -ms-grid-row: 2;
     26. -ms-grid-column: 1;
     27. margin-left: 0px;
     28. margin-top: 12px;
     29. }
     31. .itemdetailpage .content article .ingredients .item-ingredients {
     32. margin-top: 4px;
     33. }
     34. .itemdetailpage .content article .ingredients .item-ingredients .ingredient {
     35. padding-bottom: 0px;
     36. font-size: 9pt;
     37. }
     38. }
  4. Run the application again. Snap it so that it occupies the left side of the screen alongside another app. The go to the item-detail page and confirm that the snapped page now shows recipe ingredients, as pictured in Figure 11.
     1. 
     2. Figure 11
     3. The modified item-detail page in snapped mode
  5. Return to Visual Studio and stop debugging.

Exercise 3: Semantic Zoom

* 1. Many applications that run on touch screens allow users to zoom in and out using a two-fingered pinching motion. A photo-editing application, for example, might let you zoom in on a photo when two fingers touching the screen move apart, and zoom back out when two fingers touching the screen move together.
  2. Most zooms are optical zooms, meaning they simply scale content displayed on the screen. It is easy enough to include optical zoom in a Metro style app, but Windows 8 also supports *semantic zoom*. Semantic zoom does not simply scale content up or down; it groups the content to provide a different semantic view. For example, semantic zoom in a mapping application might reveal additional details such as street names and building names as the user zooms in, and remove those details when the user zooms back out.
  3. To aid you in implementing semantic zoom, WinJS includes a SemanticZoom control. The basic idea is that you provide the control with two views – a zoomed-in view and a zoomed-out view – and the control will switch between the two in response to user input. You do not have to do the switching yourself, and you do not have to write gesture-recognition code to take action when two fingers make contact with the screen and move together or apart. If you do not have a touch screen, you can zoom in and out by holding down the Ctrl key and rolling the mouse wheel, or tapping Ctrl-Plus and Ctrl-Minus.   
     Sound appealing? Then let’s put semantic zoom to work.

Task 1 – Add a SemanticZoom Control to the Start Page

* 1. The first step in adding semantic zoom to Contoso Cookbook is to declare a SemanticZoom control and to expand the one ListView currently declared in the start page to two ListViews.

1. Open groupedItems.html.
2. Find the SECTION element containing a ListView control at the bottom. Rewrite the SECTION element to include a SemanticZoom control and two ListView controls: one representing a zoomed-in view of the start page, and another representing a zoomed-out view:
   * 1. HTML
     2. <section aria-label="Main content" role="main">
     3. <div id="zoom" data-win-control="WinJS.UI.SemanticZoom" data-win-options="{ initiallyZoomedOut: false }" style="height: 100%">
     4. <div id="zoomedInListView" class="groupeditemslist" aria-label="List of groups" data-win-control="WinJS.UI.ListView" data-win-options="{ selectionMode: 'none' }"></div>
     5. <div id="zoomedOutListView" class="groupeditemslist" aria-label="List of groups" data-win-control="WinJS.UI.ListView" data-win-options="{ selectionMode: 'none' }"></div>
     6. </div>
     7. </section>
     8. **Note:** We are adding semantic zoom to the start page so the user can zoom out and see all the recipe groups on one screen. The start page as it currently exists will become the zoomed-in view, and setting the SemanticZoom control’s initiallyZoomedOut property to false ensures that the user initially sees the zoomed-in view.
3. Add the following statements to groupedItems.html after the DIV whose class is “itemtemplate:”
   * 1. HTML
     2. <div class="zoomedOutItemTemplate" data-win-control="WinJS.Binding.Template">
     3. <div class="zoomedOutItemContainer" >
     4. <img class="zoomeOutItemImage" src="#" data-win-bind="src: groupImage; alt: title" />
     5. <h4 class="zoomedOutGroupTitle" data-win-bind="textContent: title; alt: title"> </h4>
     6. <div class="zoomedOutItemCount" data-win-bind="textContent: recipesCount;"></div>
     7. </div>
     8. </div>

Task 2 – Modify groupedItems.js

* 1. Now we need to make some changes to have the right data drive our semantic zoom view.
  2. Open groupedItems.js.
  3. Replace the ready function and the updateLayout function with the ones below:
     1. JavaScript
     2. // This function is called whenever a user navigates to this page. It
     3. // populates the page elements with the app's data.
     4. ready: function (element, options) {
     5. var semanticZoom = element.querySelector("#zoom").winControl;
     6. var zoomedInListView = element.querySelector("#zoomedInListView").winControl;
     7. var zoomedOutListView = element.querySelector("#zoomedOutListView").winControl;
     8. zoomedOutListView.itemTemplate = element.querySelector(".zoomedOutItemTemplate");
     9. zoomedOutListView.itemDataSource = Data.groups.dataSource;
     10. zoomedOutListView.groupDataSource = null;
     11. zoomedOutListView.layout = new ui.GridLayout({ groupHeaderPosition: "top" });
     12. zoomedInListView.groupHeaderTemplate = element.querySelector(".headerTemplate");
     13. zoomedInListView.itemTemplate = element.querySelector(".itemtemplate");
     14. zoomedInListView.oniteminvoked = this.itemInvoked.bind(this);
     15. if (appView.value === appViewState.snapped) {
     16. // If the app is snapped, configure the zoomed-in ListView
     17. // to show groups and lock the SemanticZoom control
     18. zoomedInListView.itemDataSource = Data.groups.dataSource;
     19. zoomedInListView.groupDataSource = null;
     20. zoomedInListView.layout = new ui.ListLayout();
     21. semanticZoom.locked = true;
     22. }
     23. else {
     24. // If the app isn't snapped, configure the zoomed-in ListView
     25. // to show items and groups and unlock the SemanticZoom control
     26. zoomedInListView.itemDataSource = Data.items.dataSource;
     27. zoomedInListView.groupDataSource = Data.groups.dataSource;
     28. zoomedInListView.layout = new ui.GridLayout({ groupHeaderPosition: "top" });
     29. semanticZoom.locked = false;
     30. }
     31. semanticZoom.element.focus();
     32. },
     33. // This function updates the page layout in response to viewState changes.
     34. updateLayout: function (element, viewState, lastViewState) {
     35. /// <param name="element" domElement="true" />
     36. /// <param name="viewState" value="Windows.UI.ViewManagement.ApplicationViewState" />
     37. /// <param name="lastViewState" value="Windows.UI.ViewManagement.ApplicationViewState" />
     38. var semanticZoom = element.querySelector("#zoom").winControl;
     39. var zoomedInListView = element.querySelector("#zoomedInListView").winControl;
     40. if (appView.value === appViewState.snapped) {
     41. zoomedInListView.itemDataSource = Data.groups.dataSource;
     42. zoomedInListView.groupDataSource = null;
     43. zoomedInListView.layout = new ui.ListLayout();
     44. semanticZoom.zoomedOut = false;
     45. semanticZoom.locked = true;
     46. }
     47. else {
     48. zoomedInListView.itemDataSource = Data.items.dataSource;
     49. zoomedInListView.groupDataSource = Data.groups.dataSource;
     50. zoomedInListView.layout = new ui.GridLayout({ groupHeaderPosition: "top" });
     51. semanticZoom.zoomedOut = true;
     52. semanticZoom.locked = false;
     53. // Shouldn't be necessary, but without this statement, the
     54. // ListView wants to scroll vertically when the app is unsnapped
     55. // after being snapped while zoomed out
     56. nav.navigate("/pages/groupedItems/groupedItems.html");
     57. }
     58. }

Task 3 – Modify groupedItems.css

* 1. You are almost ready to see semantic zoom in action. First, we need to style the zoomed-out ListView control.
  2. Open groupedItems.css.
  3. Add the following statements at the top:
     1. CSS
     2. #zoomedOutListView {
     3. margin-left: 75px;
     4. }
     5. #zoomedOutListView .win-item {
     6. height: 500px;
     7. width: 250px;
     8. }
     9. #zoomedOutListView .win-container {
     10. margin-right:30px;
     11. }
     12. .zoomedOutItemContainer {
     13. min-height: 500px;
     14. height: 100%;
     15. width: 250px;
     16. display: -ms-grid;
     18. -ms-grid-rows: 1fr 130px 400px 1fr;
     19. -ms-grid-columns: 1fr;
     20. }
     21. .zoomeOutItemImage {
     22. -ms-grid-row-span: 4;
     23. }
     24. .zoomedOutGroupImage {
     25. -ms-grid-row: 3;
     26. }
     27. .zoomedOutItemCount {
     28. -ms-grid-row: 2;
     29. -ms-grid-column-align: end;
     30. font-size: 80px;
     31. color: rgb(250, 243, 214);
     32. margin: -7px 12px 0px 20px;
     33. font-weight: 100;
     34. }
     35. .zoomedOutGroupTitle {
     36. -ms-grid-row: 2;
     37. font-size: 32px;
     38. color: rgb(250, 243, 214);
     39. margin: 7px 0px 0px 16px;
     40. font-weight: 200;
     41. }
  4. Run the application and confirm that you see the same start page you’ve seen before.
  5. Put two fingers on the screen and move them together (or press the Ctrl key and the - key) to zoom out. Verify that the page changes to the one shown in Figure 12.
     1. 
     2. Figure 12
     3. The zoomed-out start page
  6. Put two fingers on the screen again, but this time move them away from each other (or use the Ctrl key and the + key) to zoom in. What happens to the start page?
  7. Zoom out again and tap one of the recipe groups. Verify that you switch back to the zoomed-in view, and that it’s scrolled to the group you selected.
  8. Return to Visual Studio and stop debugging.

Summary

* 1. In this lab, you made some key UI-related enhancements to Contoso Cookbook. You adapted the layout for rotated displays, optimized the item-detail page to show additional recipe data when the application is snapped, and added semantic zoom to simplify the task of navigating among recipe groups on the start page. And you did all of this while writing remarkably little code.
  2. Most of the work in Labs 1 and 2 involved creating the user experience – getting the pages looking just right in all orientations and all modes. Now it is time to do some work on the back end by adding support for sharing and search. Contoso Cookbook is about to become more tightly integrated with Windows and other Metro style apps.