Hands-On Lab

Orientation, Snapping, and Semantic Zoom

Lab version: 1.0.0

Last updated: 6/9/2012



Contents

[Overview 3](#_Toc320344766)

[Exercise 1: Device Orientation 4](#_Toc320344767)

[Task 1 – Test the Start Page 4](#_Toc320344768)

[Task 2 – Customize the Group-Detail Page 7](#_Toc320344769)

[Task 3 – Customize the Item-Detail Page 10](#_Toc320344770)

[Exercise 2: Snapping 15](#_Toc320344771)

[Task 1 – Run Contoso Cookbook in Snapped Mode 16](#_Toc320344772)

[Task 2 – Modify the Snapped Item-Detail Page 17](#_Toc320344773)

[Exercise 3: Semantic Zoom 19](#_Toc320344774)

[Task 1 – Add a SemanticZoom Control to the Start Page 20](#_Toc320344775)

[Task 2 – Test the Results 22](#_Toc320344776)

[Summary 22](#_Toc320344777)

Overview

* 1. One of the hallmarks of a great Metro-style app is that it adapts to screens of various sizes, and even adapts when the screen size changes because the host device is rotated between landscape mode and portrait mode. Something else that can alter the amount of screen space available to an application is *snapping*, which allows two Metro applications 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 content itself 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’ll build upon Lab 1 by adding three important UI-related features to Contoso Cookbook. First, you’ll customize the layout of the item-detail and group-detail pages when the screen is rotated. Second, you’ll customize the layout of the item-detail page when the application is snapped. Finally, you’ll 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 ~~Consumer~~Release Preview
  + Microsoft Visual Studio 2011 Beta for Windows 8

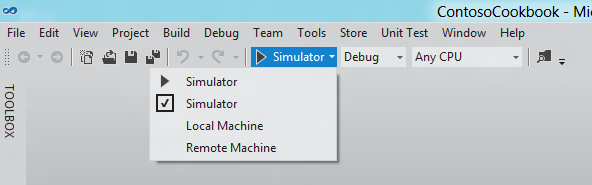
# Setup

* 1. You must perform the following steps to prepare your computer for this lab:
  2. Install the Microsoft Windows 8 ~~Consumer~~Release Preview
  3. Install the Microsoft Visual Studio 2011 Beta for Windows 8

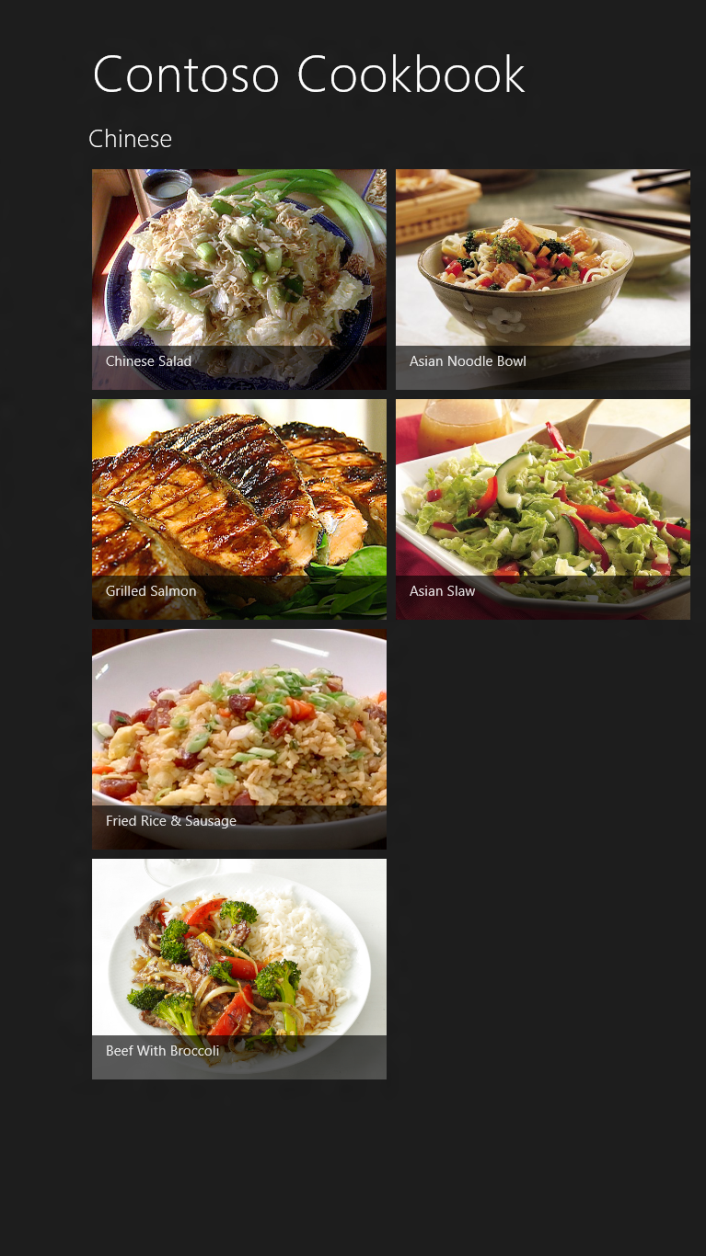
# Exercises

* 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 60minutes**.

Exercise 1: Device Orientation

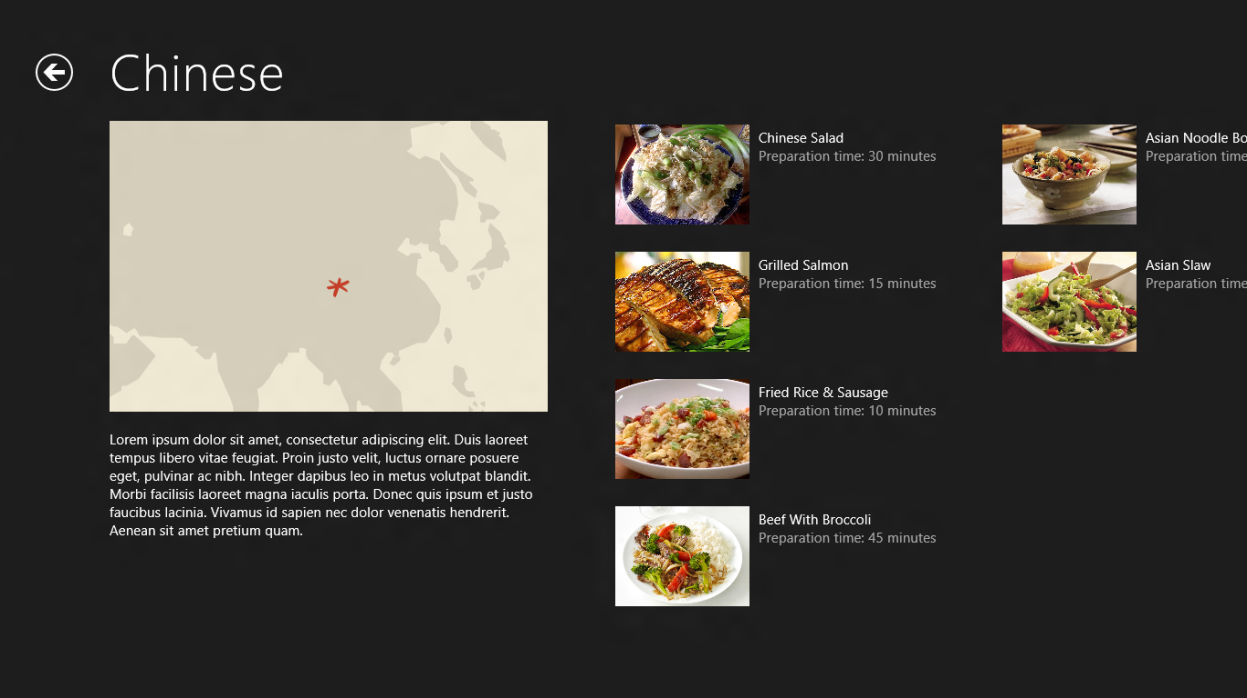
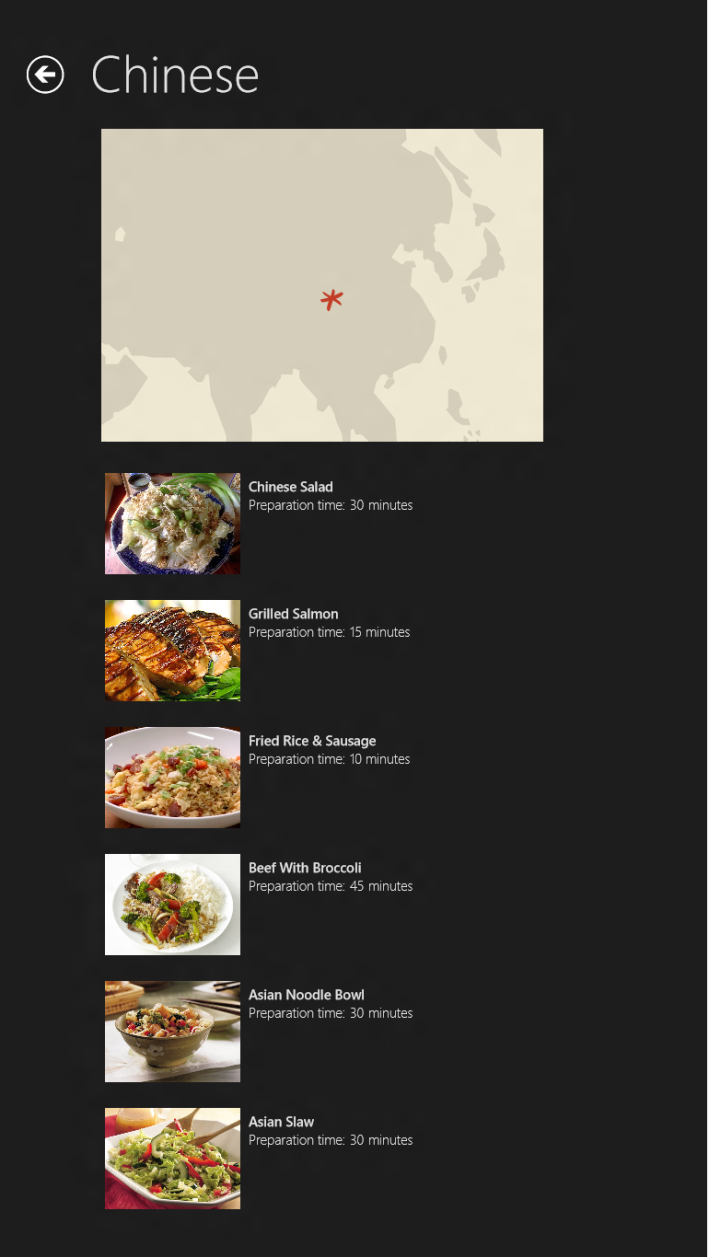
1. Windows 8 is designed to run on a variety of devices, including tablets and other devices that, through the aid of onboard accelerometers, know whether they’re in landscape mode or portrait mode. A page that looks great in landscape mode may need tweaking to look equally great in portrait mode, and vice versa. In this exercise, you’ll modify Contoso Cookbookto adapt to device orientation.
   1. **Note:** You don’t 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 byexamining 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 didn’t complete Lab 1 or would like to start with a reference copy, you’ll 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.
  4. 
  5. Figure 1
  6. The start page in landscape mode
  7. Now rotate the device to portrait mode. Make sure the screen rotates 90 degrees, and that the start page rotates as well, assuming the layout shown in Figure 2.
     1. **Note:** If you’re testing on a tablet and the start page doesn’t 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.
     2. If the hardware you’re testing on doesn’t support orientation changes, remember that you can use the Windows Simulator instead. To rotate the simulator, click either of the buttons pointed to by the red arrow in the image below.
     3. 
  8. 
  9. Figure 2
  10. The start page in portrait mode
  11. Thanks to the GridView control that renders the recipe items, the start page looks fine in both landscape and portrait modes, so no additional work is required here.
  12. Return to the Visual Studio and stop debugging.

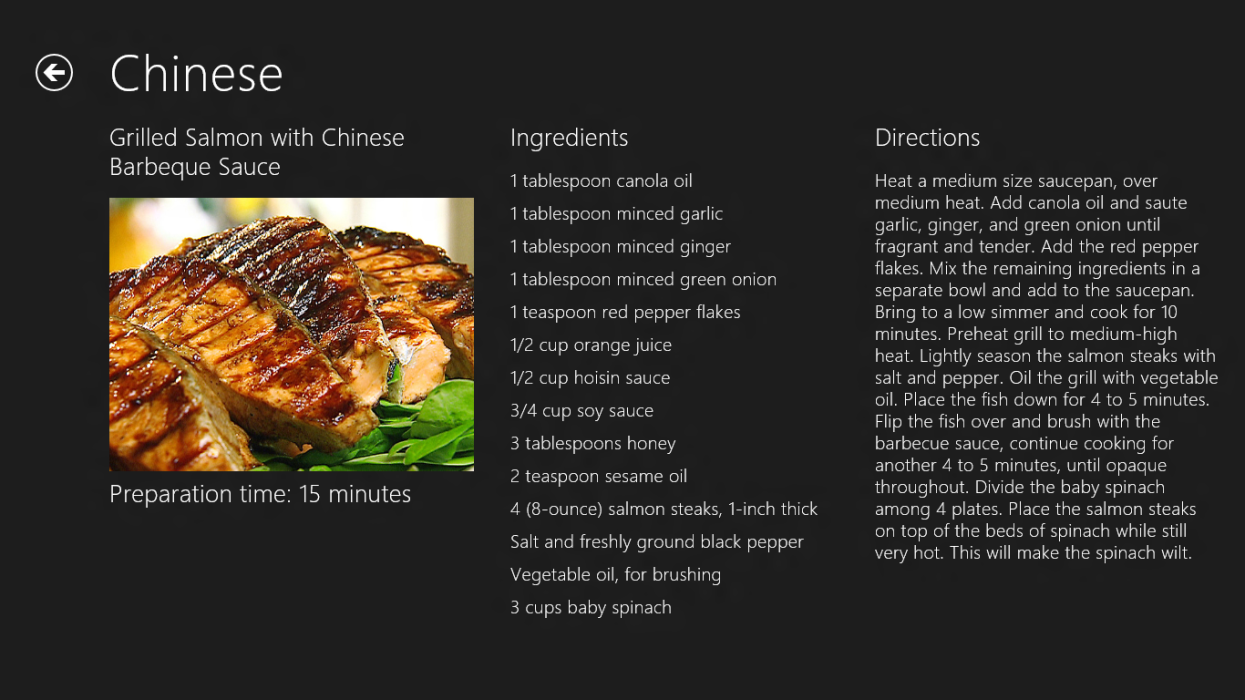
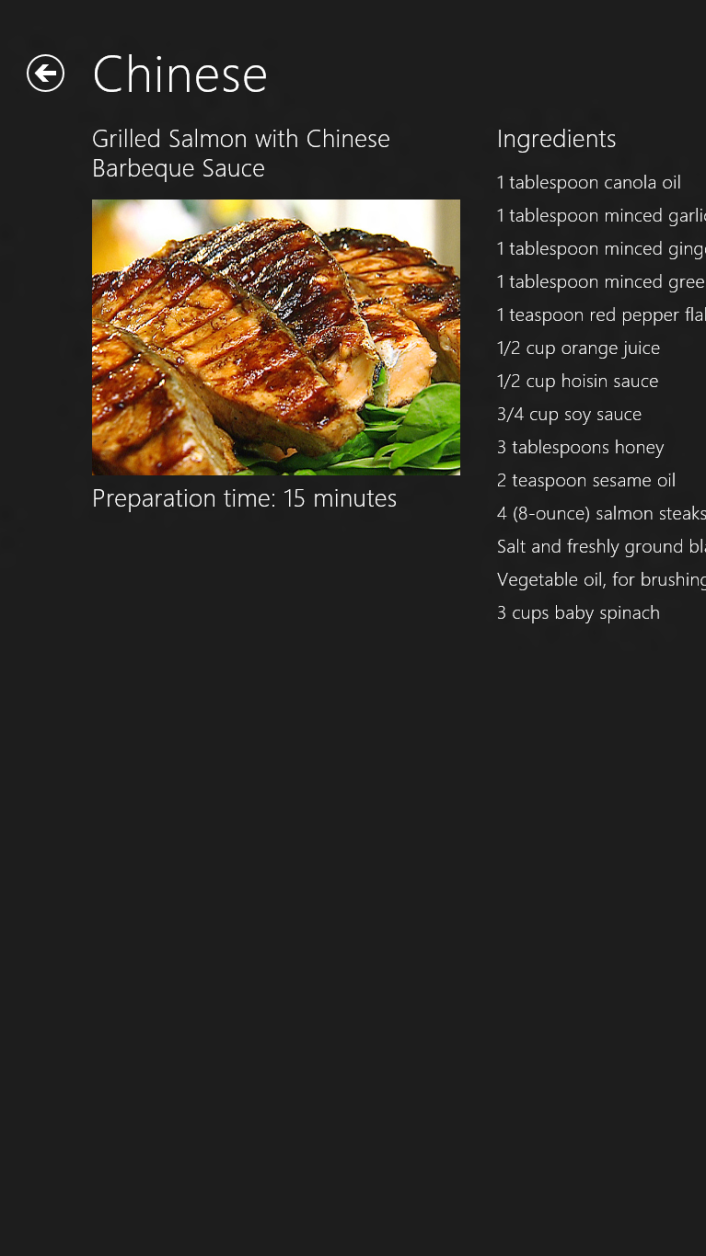
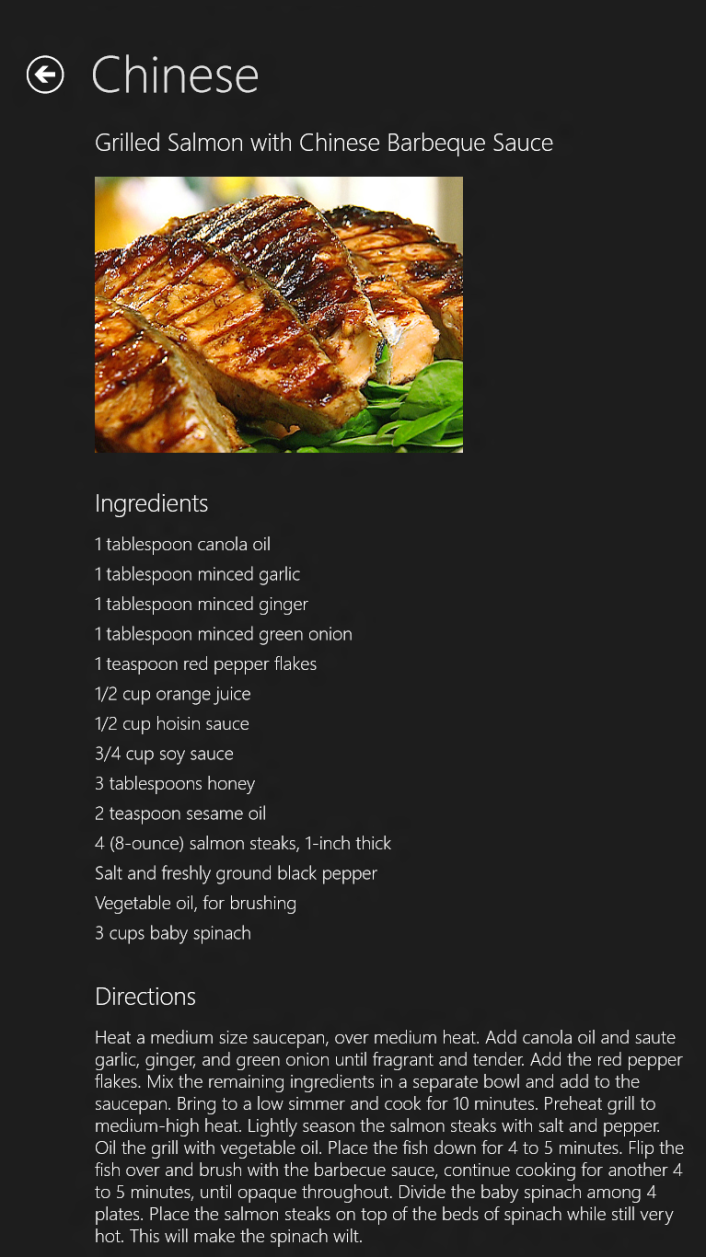
Task 2 –Customize the Group-Detail Page

The Contoso Cookbook application comprises three pages: the start page, the group-detail page, and the item-detail page. The start page doesn’t 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.”

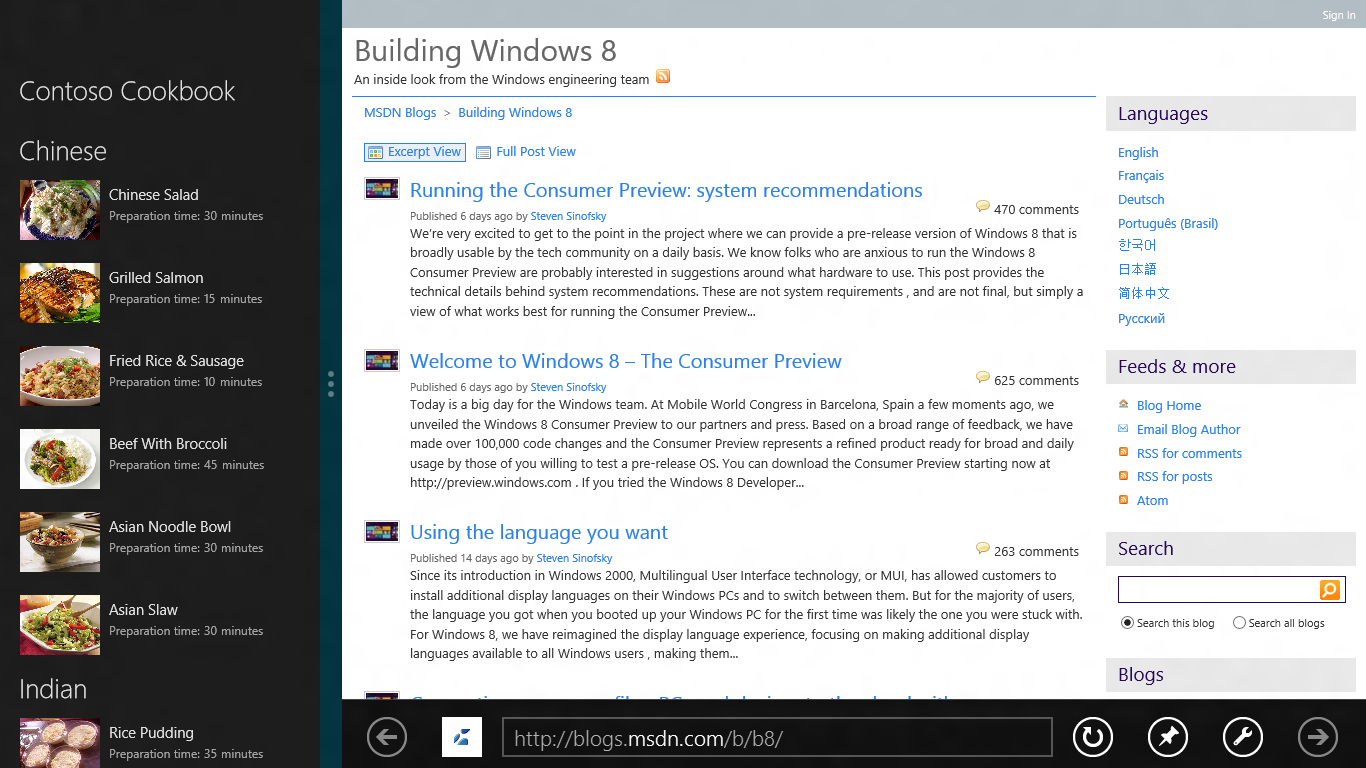
* 1. Start the application again and tap “Chinese” to show the Chinese group-detail page. In landscape mode, the page assumes the layout in Figure 3.
  2. 
  3. Figure 3
  4. The group-detail page in landscape mode
  5. Now rotate the device to portrait mode (see Figure 4). Observe that the GridView 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.
  6. 
  7. Figure 4
  8. The group-detail page in portrait mode
  9. Return to Visual Studio and stop debugging.
  10. Open GroupDetailPage.xaml and find the ListViewwhose name is “itemListView.”
  11. Immediately after “itemListView,” add ScrollViewer named “portraitScrollViewer:”
      1. XAML
      2. <!-- Vertical scrolling list used in portrait mode -->
      3. <ScrollViewer
      4. x:Name="portraitScrollViewer"
      5. AutomationProperties.AutomationId="SnappedDetailsScrollViewer"
      6. Grid.Row="1"
      7. Visibility="Collapsed"
      8. Style="{StaticResource VerticalScrollViewerStyle}">
      9. <StackPanel Margin="90,0,90,0">
      10. <Image Source="{Binding Image}" Width="480" HorizontalAlignment="Left" Margin="20,-20,0,20" Stretch="UniformToFill"/>
      11. <ListView
      12. AutomationProperties.AutomationId="ItemListView"
      13. AutomationProperties.Name="Items In Group"
      14. Margin="10,0,0,60"
      15. ItemsSource="{Binding Source={StaticResource itemsViewSource}}"
      16. ItemTemplate="{StaticResource Standard500x130ItemTemplate}"
      17. SelectionMode="None"
      18. IsItemClickEnabled="True"
      19. ItemClick="ItemView\_ItemClick"/>
      20. </StackPanel>
      21. </ScrollViewer>
  12. Scroll further down in GroupDetailPage.xaml and find the VisualState element whose name is “FullScreenPortrait.”
  13. Add the following ObjectAnimationUsingKeyFrames elements to the Storyboard inside the VisualState element, after the existing ObjectAnimationUsingKeyFrames elements:
      1. XAML
      2. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="gridScrollViewer" Storyboard.TargetProperty="Visibility">
      3. <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
      4. </ObjectAnimationUsingKeyFrames>
      5. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="portraitScrollViewer" Storyboard.TargetProperty="Visibility">
      6. <DiscreteObjectKeyFrame KeyTime="0" Value="Visible"/>
      7. </ObjectAnimationUsingKeyFrames>
      8. **Note:** The ScrollViewer you added defines how recipe groups are rendered in portrait mode. The animations display that ScrollViewer and hide the default ScrollViewer when the view state changes to “FullScreenPortrait.” (Metro view state has nothing to do with ASP.NET view state, so don’t 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. The view state also changes when the application is “snapped.” Snapping is covered in the next exercise.
  14. Run the application, tap “Chinese” to display the group-detail page, and rotate the device to portrait mode. Confirm that the page looks like the one in Figure 5.
  15. 
  16. Figure 5
  17. The finished portrait-mode layout
  18. Return to Visual Studio and stop debugging.

Task 3 –Customize the Item-Detail Page

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.

* 1. Start the application again and tap “Steam Bun Baos” to show the recipe-detail page. Figure 6 shows how the page looks in landscape mode.
  2. 
  3. Figure 6
  4. The item-detail page in landscape mode
  5. Now rotate the device to portrait mode (Figure 7). Once more, the layout that works well in landscape mode needs a bit of tweaking for portrait mode.
  6. 
  7. Figure 7
  8. The item-detail page in portrait mode
  9. Return to Visual Studio and stop debugging.
  10. Open ItemDetailPage.xaml and find the FlipView control named “flipView.”
  11. After that FlipView (that is, after the ending </FlipView> tag), add another FlipView control:
      1. XAML
      2. <!-- FlipView used in portrait mode -->
      3. <FlipView
      4. x:Name="portraitFlipView"
      5. AutomationProperties.AutomationId="ItemsFlipView"
      6. AutomationProperties.Name="Item Details"
      7. Grid.Row="1"
      8. Margin="0,-3,0,0"
      9. ItemsSource="{Binding Source={StaticResource itemsViewSource}}"
      10. Visibility="Collapsed">
      11. <FlipView.ItemTemplate>
      12. <DataTemplate>
      13. <UserControl Loaded="StartLayoutUpdates" Unloaded="StopLayoutUpdates">
      14. <ScrollViewer x:Name="scrollViewer" Style="{StaticResource VerticalScrollViewerStyle}" Grid.Row="1">
      15. <!-- Vertical StackPanel for item-detail layout -->
      16. <StackPanel Orientation="Vertical" Margin="100,0,20,0">
      17. <StackPanel Orientation="Vertical">
      18. <TextBlock FontSize="26.667" FontWeight="Light" Text="{Binding Title}" TextWrapping="Wrap"/>
      19. <Image x:Name="image" Width="400" Margin="0,20,0,40" Stretch="Uniform" Source="{Binding Image}" HorizontalAlignment="Left"/>
      20. </StackPanel>
      21. <StackPanel Orientation="Vertical">
      22. <TextBlock FontSize="26.667" FontWeight="Light" Text="Ingredients" Margin="0,0,0,16"/>
      23. <TextBlock FontSize="20" FontWeight="Light" LineHeight="32.5" Text="{Binding Ingredients, Converter={StaticResource ListConverter}}" TextWrapping="Wrap" />
      24. </StackPanel>
      25. <StackPanel Orientation="Vertical">
      26. <TextBlock FontSize="26.667" FontWeight="Light" Text="Directions" Margin="0,0,0,16"/>
      27. <ScrollViewer Style="{StaticResource VerticalScrollViewerStyle}">
      28. <Grid>
      29. <TextBlock FontSize="20" FontWeight="Light" Text="{Binding Directions}" TextWrapping="Wrap" />
      30. </Grid>
      31. </ScrollViewer>
      32. </StackPanel>
      33. </StackPanel>
      34. </ScrollViewer>
      35. </UserControl>
      36. </DataTemplate>
      37. </FlipView.ItemTemplate>
      38. </FlipView>
      39. **Note:** In landscape mode, the three content areas in the item-detail page – the recipe name, image, and description; the ingredients; and the directions – are laid out in columns of a multi-column Grid declared in the data template of the FlipView control named “flipView.” The FlipView you just added is for portrait mode. It uses a vertical StackPanel to stack the content areas on top of each other.
  12. Find the VisualState element whose name is “FullScreenPortrait.”
  13. Add the following statements to the Storyboard inside the VisualState element to switch from the “flipView” control to the “portraitFlipView” control you just added when the screen rotates to portrait mode:
      1. XAML
      2. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="flipView" Storyboard.TargetProperty="Visibility">
      3. <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
      4. </ObjectAnimationUsingKeyFrames>
      5. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="portraitFlipView" Storyboard.TargetProperty="Visibility">
      6. <DiscreteObjectKeyFrame KeyTime="0" Value="Visible"/>
      7. </ObjectAnimationUsingKeyFrames>
  14. Start the application and tap “Steam Bun Baos” again to show the recipe-detail page. Rotate the display to portrait mode and verify that it assumes the single-column layout depicted in Figure 8.
  15. 
  16. Figure 8
  17. The modified portrait-mode layout
  18. Rotate back to landscape mode and make sure the page reverts to a 3-column layout.
  19. Return to Visual Studio and stop debugging.

Exercise 2: Snapping

1. Snapping enables Windows 8 users to run two Metro style applications 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 don’t have a touch screen, you can press the Windows key and the period key instead.) Figure 9 shows snapping in action. Contoso Cookbookoccupies the left 320 pixels of the screen, while Internet Explorer occupies the remaining portion. In this context, Contoso Cookbookis the “snapped app” and Internet Explorer is the “main app.” If desired, the user can drag the snap bar across the screen and reverse the apps’ roles.
   1. 
   2. Figure 9
   3. Snapping in action

Contoso Cookbook already has some snapping behavior built in, thanks to the XAML Visual Studio included in the pages it generated for the project. In this exercise, you’ll apply a simple change to one of those pages 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 1,366 by 768 pixels. The Metro team chose 1,366 as the minimum because it affords the snapped app a section of the screen that’s 320 pixels wide (the same width as many smartphones) and the main app a section that’s 1,024 pixels wide. The extra pixels areconsumed by the snap bar. If you’re running on a lower resolution screen, use the Windows Simulator for this exercise and select a simulated screen resolution of at least 1,366 by 768.

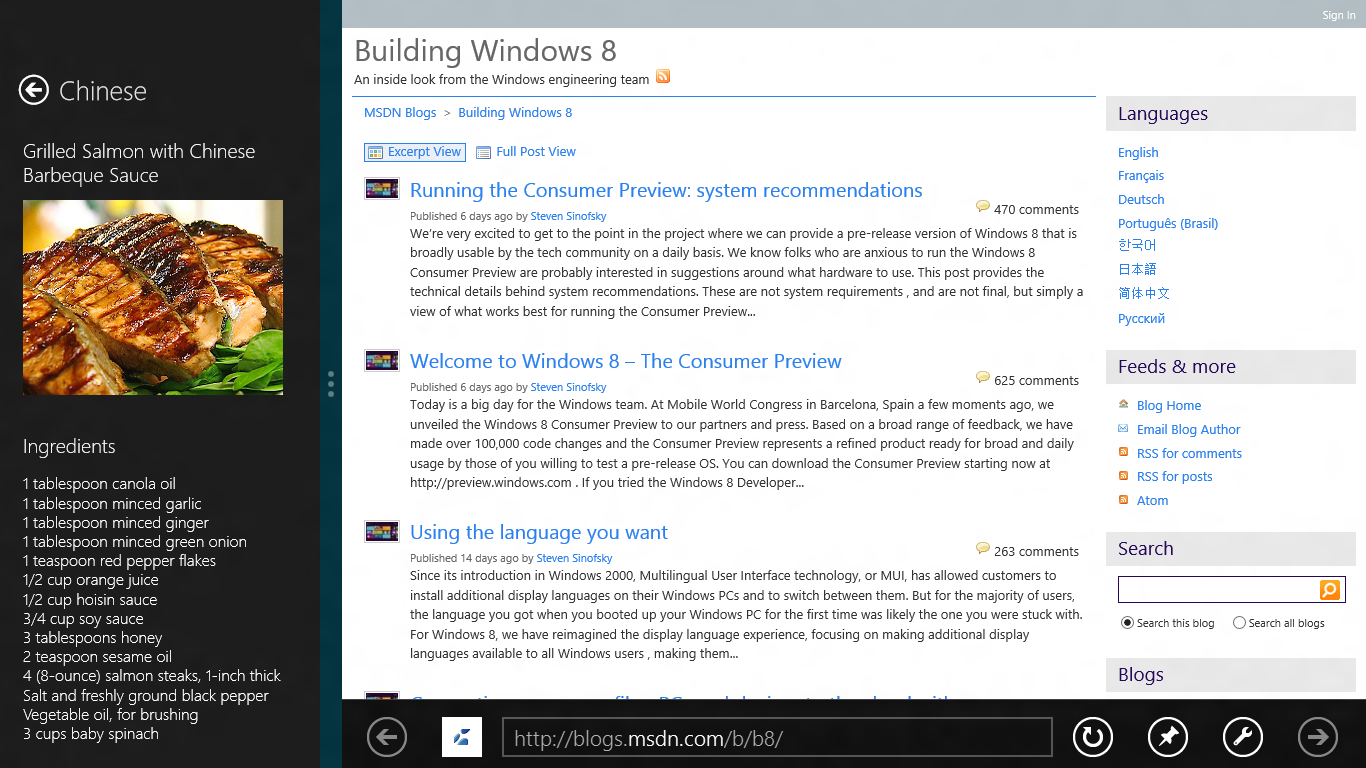
Task 1 – Run Contoso Cookbook in Snapped Mode

In this task, you’ll view each of Contoso Cookbook’s three pages when the application is snapped.

* 1. 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.
  2. When Contoso Cookbook only occupies a portion of the screen, how does its start page differ from when it’s displayed full-screen?
  3. Tap one of the group names (for example, “Chinese”) on the start page to see the group-detail page in snapped mode. How does it differ from its full-screen equivalent?
  4. Tap one of the recipes to see the item-detail page in snapped mode. How does its content and layout compare to when it’s running full screen?
  5. Return to Visual Studio and stop debugging.

Task 2 – Modify the Snapped Item-Detail Page

The default snapped layouts provided by Visual Studio were a great start, but you can further customize them if desired. 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.

* 1. Open ItemDetailPage.xaml and add the following FlipView control to the FlipViews that are already there:
     1. XAML
     2. <FlipView
     3. x:Name="snappedFlipView"
     4. AutomationProperties.AutomationId="ItemsFlipView"
     5. AutomationProperties.Name="Item Details"
     6. Grid.Row="1"
     7. Margin="0,-3,0,0"
     8. ItemsSource="{Binding Source={StaticResource itemsViewSource}}"
     9. Visibility="Collapsed">
     10. <FlipView.ItemTemplate>
     11. <DataTemplate>
     12. <UserControl Loaded="StartLayoutUpdates" Unloaded="StopLayoutUpdates">
     13. <ScrollViewer x:Name="scrollViewer" Style="{StaticResource VerticalScrollViewerStyle}" Grid.Row="1">
     14. <!-- Vertical StackPanel for item-detail layout -->
     15. <StackPanel Orientation="Vertical" Margin="20,0,20,0">
     16. <StackPanel Orientation="Vertical">
     17. <TextBlock FontSize="20" FontWeight="Light" Text="{Binding Title}" TextWrapping="Wrap"/>
     18. <Image x:Name="image" Width="260" Margin="0,12,0,40" Stretch="Uniform" Source="{Binding Image}" HorizontalAlignment="Left"/>
     19. </StackPanel>
     20. <StackPanel Orientation="Vertical">
     21. <TextBlock FontSize="20" FontWeight="Light" Text="Ingredients" Margin="0,0,0,16"/>
     22. <TextBlock FontSize="16" FontWeight="Light" TextWrapping="Wrap" Text="{Binding Ingredients, Converter={StaticResource ListConverter}}" />
     23. </StackPanel>
     24. </StackPanel>
     25. </ScrollViewer>
     26. </UserControl>
     27. </DataTemplate>
     28. </FlipView.ItemTemplate>
     29. </FlipView>
     30. **Note:** This is the second FlipView you’ve added to the item-detail page. The first one handles portrait-mode layout. This one will be used only when the page is snapped.
  2. Find the VisualState element named “Snapped” near the bottom of ItemDetailPage.xaml.
  3. Add the following animations to the animations already present in the VisualState element’s Storyboard to hide the default FlipView and show the snapped FlipView when the view state changes to “Snapped:”
     1. XAML
     2. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="flipView" Storyboard.TargetProperty="Visibility">
     3. <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
     4. </ObjectAnimationUsingKeyFrames>
     5. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="snappedFlipView" Storyboard.TargetProperty="Visibility">
     6. <DiscreteObjectKeyFrame KeyTime="0" Value="Visible"/>
     7. </ObjectAnimationUsingKeyFrames>
  4. Start 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 10.
  5. 
  6. Figure 10
  7. The modified item-detail page in snapped mode
  8. 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’s easy enough to include optical zoom in a Metro-style application, but Windows 8 also supports *semantic zoom*. Semantic zoom doesn’t 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, WinRT’s Windows.UI.Xaml.Controls namespace 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 let the control switch between the two in response to user input. You don’t have to do the switching yourself, and you don’t have to write gesture-recognition code to take action when two fingers make contact with the screen and move together or apart. And if you don’t have a touch screen, you can zoom in and out by holding down the Ctrl key and rolling the mousewheel. Sound appealing? Then let’s put semantic zoom to work.

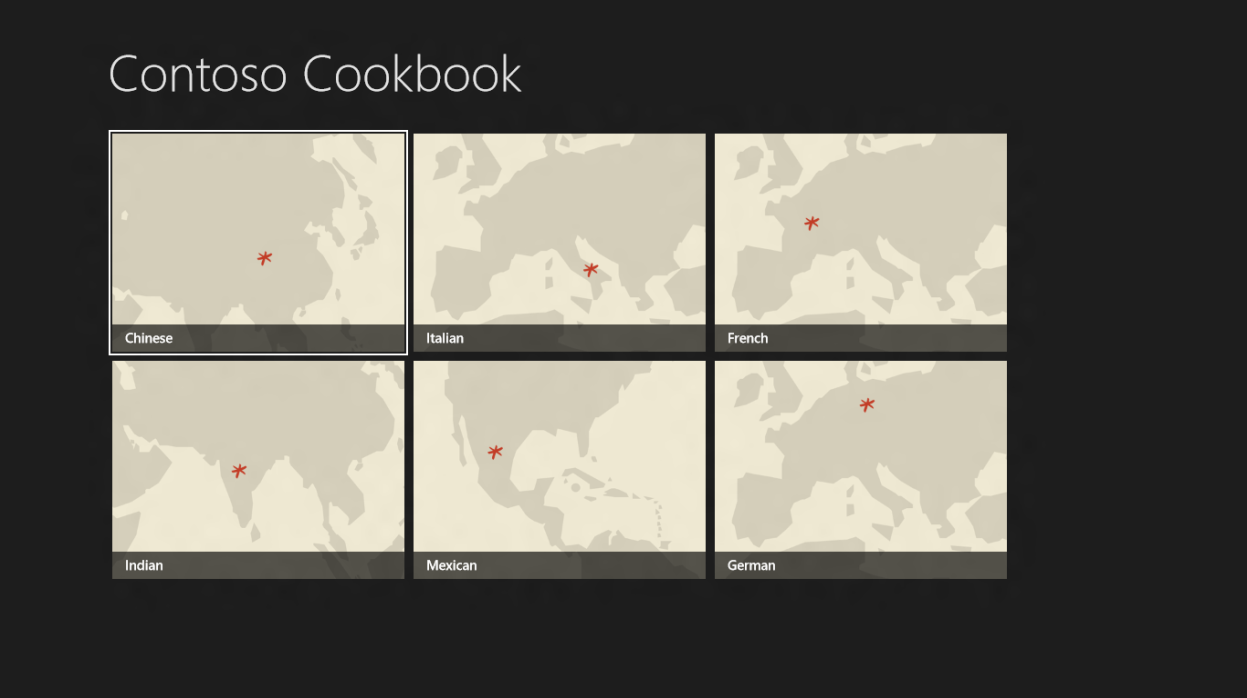
Task 1 –Add a SemanticZoom Control to the Start Page

Adding semantic zoom to Contoso Cookbook is as simple as declaring a SemanticZoom control and populating it with views representing the zoomed-in view of the start page and the zoomed-out view.

* 1. Open GroupedItemsPage.xaml.
  2. ~~Find the ScrollViewer named “itemGridScrollViewer.” Delete it and everything inside it.~~
  3. Replace the ScrollViewer you just deleted with the following statements:
     1. XAML
     2. <SemanticZoom Grid.Row="1">
     3. <SemanticZoom.ZoomedInView>
     4. <GridView
     5. x:Name="itemGridView"
     6. AutomationProperties.AutomationId="ItemGridView"
     7. AutomationProperties.Name="Grouped Items"
     8. ItemsSource="{Binding Source={StaticResource groupedItemsViewSource}}"
     9. ItemTemplate="{StaticResource Standard250x250ItemTemplate}"
     10. SelectionMode="None"
     11. IsItemClickEnabled="True"
     12. ItemClick="ItemView\_ItemClick">
     13. <GridView.ItemsPanel>
     14. <ItemsPanelTemplate>
     15. <VirtualizingStackPanel Orientation="Horizontal" Margin="116,0,40,46" />
     16. </ItemsPanelTemplate>
     17. </GridView.ItemsPanel>
     18. <GridView.GroupStyle>
     19. <GroupStyle>
     20. <GroupStyle.HeaderTemplate>
     21. <DataTemplate>
     22. <Grid Margin="1,0,0,6">
     23. <Button
     24. AutomationProperties.Name="Group Title"
     25. Content="{Binding Title}"
     26. Click="Header\_Click"
     27. Style="{StaticResource TextButtonStyle}"/>
     28. </Grid>
     29. </DataTemplate>
     30. </GroupStyle.HeaderTemplate>
     31. <GroupStyle.Panel>
     32. <ItemsPanelTemplate>
     33. <VariableSizedWrapGrid Orientation="Vertical" Margin="0,0,80,0"/>
     34. </ItemsPanelTemplate>
     35. </GroupStyle.Panel>
     36. </GroupStyle>
     37. </GridView.GroupStyle>
     38. </GridView>
     39. </SemanticZoom.ZoomedInView>
     40. <SemanticZoom.ZoomedOutView>
     41. <GridView x:Name="groupGridView" Margin="116,0,40,46">
     42. <GridView.ItemTemplate>
     43. <DataTemplate>
     44. <Grid>
     45. <Image Source="{Binding Group.Image}" Width="320" Height="240" Stretch="UniformToFill" />
     46. <StackPanel VerticalAlignment="Bottom" Background="{StaticResource ListViewItemOverlayBackgroundBrush}">
     47. <TextBlock Text="{Binding Group.Title}" Foreground="{StaticResource ListViewItemOverlayTextBrush}" Style="{StaticResource TitleTextStyle}" Height="30" Margin="15,0,15,0"/>
     48. </StackPanel>
     49. </Grid>
     50. </DataTemplate>
     51. </GridView.ItemTemplate>
     52. </GridView>
     53. </SemanticZoom.ZoomedOutView>
     54. </SemanticZoom>
     55. **Note:** You just replaced the ScrollViewer, which contained a GridView that rendered items to the start page, with a SemanticZoom control containing two GridViews: one representing the zoomed-in view of the start page, and one representing the zoomed-out view. The SemanticZoom control does the switching between the two.
  4. Near the bottom of GroupedItemsPage.xaml, find and delete the ObjectAnimationUsingKeyFrames element that targets “itemGridView” and replace it with these:
     1. XAML
     2. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="itemGridView" Storyboard.TargetProperty="Visibility">
     3. <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
     4. </ObjectAnimationUsingKeyFrames>
     5. <ObjectAnimationUsingKeyFrames Storyboard.TargetName="groupGridView" Storyboard.TargetProperty="Visibility">
     6. <DiscreteObjectKeyFrame KeyTime="0" Value="Collapsed"/>
     7. </ObjectAnimationUsingKeyFrames>
  5. Open GroupedItemsPage.xaml.cs and add the following statement to the end of the LoadStatemethod:
     1. C#
     2. this.groupGridView.ItemsSource = this.groupedItemsViewSource.View.CollectionGroups;

Task 2 – Test the Results

Now it’s time to see semantic zoom in action.

* 1. Run the application and confirm that you see the same start page you’ve seen before.
  2. Put two fingers on the screen and move them together (or use the mousewheel and the Ctrl key)to zoom out. Verify that the page changes to the one shown in Figure 11.
  3. 
  4. Figure 11
  5. 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 mousewheel and the Ctrl key again) 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 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’s 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 the Metro environment.