Hands-On Lab

Application Bars and Media Capture

Lab version: 1.0.0

Last updated: 6/8/2012



Contents

[Overview 3](#_Toc318972810)

[Exercise 1: Add Application Bars 4](#_Toc318972811)

[Task 1 – Add an Application Bar to the Group-Detail Page 4](#_Toc318972812)

[Task 2 – Add an Application Bar to the Item-Detail Page 6](#_Toc318972813)

[Exercise 2: Add Photo Capture 8](#_Toc318972814)

[Task 1 – Use CameraCaptureUI to Capture Photos 8](#_Toc318972815)

[Task 2 – Enable Webcam Access 10](#_Toc318972816)

[Taks 3 – Test the Results 11](#_Toc318972817)

[Exercise 3: Add Video Capture 11](#_Toc318972818)

[Task 1 – Use CameraCaptureUI to Capture Video 11](#_Toc318972819)

[Task 2 – Test the Results 12](#_Toc318972820)

[Summary 13](#_Toc318972821)

Overview

* 1. The Windows Runtime’s Windows.Media.Capture namespace contains a handy set of classes for capturing media and incorporating it into a Metro-style app. In particular, the CameraCaptureUI class makes it a breeze to use a webcam to snap photos or shoot videos. It’s even cognizant of devices that include two cameras (one front-facing, the other back-facing) and provides a simple and intuitive UI for switching between the two.
  2. In this lab, you’ll enhance Contoso Cookbook by allowing users to capture photos and videos of their favorite recipes and share them with other applications. You’ll also add an application bar that provides shortcuts to these features and learn how to manage application bars in a Metro-style app.

# Objectives

* 1. This lab will show you how to:
  + Implement application bars in a Metro-style app
  + Add commands to the application bar
  + Use the Windows Runtime to snap photos
  + Use the Windows Runtime to capture videos
  + Share photos and videos using sharing contracts

# System Requirements

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

# Setup

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

# Exercises

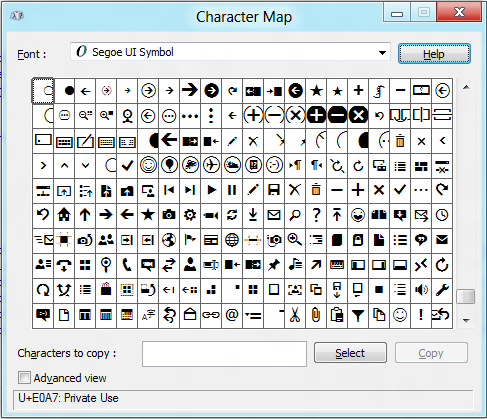
* 1. This Hands-On Lab comprises the following exercises:
  2. Add Application Bars
  3. Add Photo Capture
  4. Add Video Capture
  5. Estimated time to complete this lab: **30 to 40 minutes**.

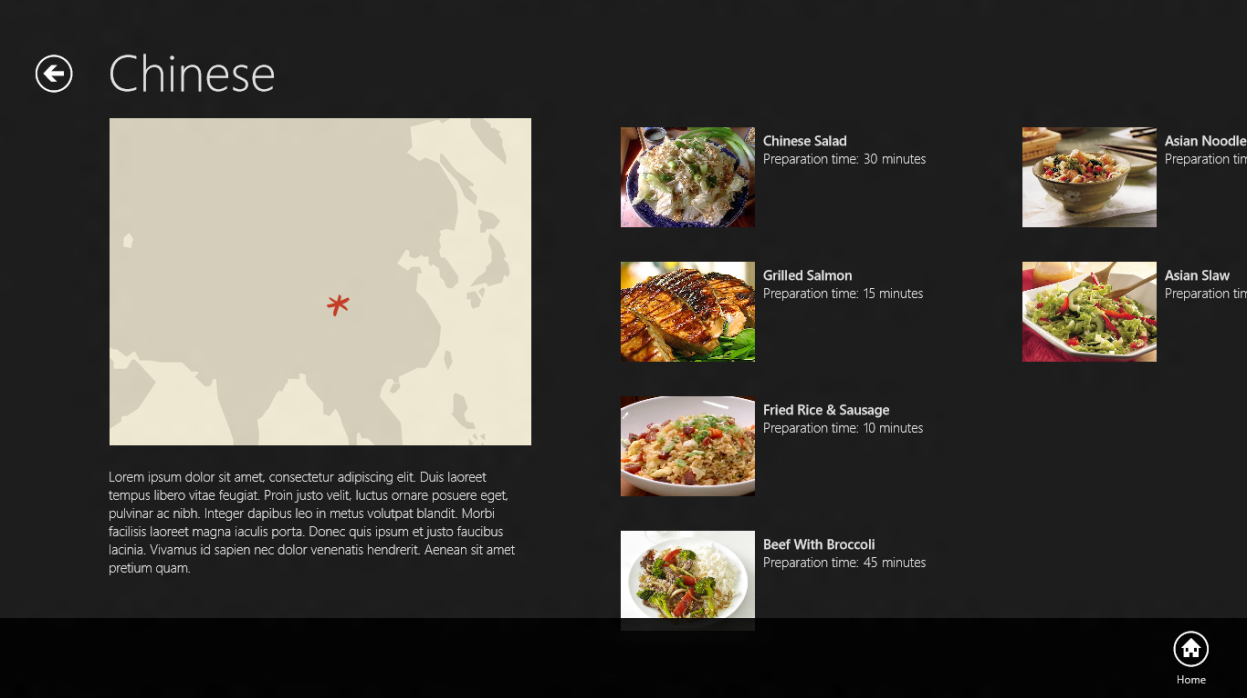
Exercise 1: Add Application Bars

1. Before we add features allowing Contoso Recipe users to capture and share photos and videos, we need to modify the user interface to provide access to those features. An application bar is the perfect tool for the job. We’ll add one application bar containing a Home command to the group-detail page, and another one containing additional commands to the item-detail page.

Task 1 – Add an Application Bar to the Group-Detail Page

* 1. We’ll start by modifying GroupDetailPage.xaml to include a simple application bar.
  2. Open the ContosoCookbook project you finished in Lab 3 in Visual Studio. If you didn’t complete Lab 3 or would like to start with a reference copy, you’ll find a completed version of the lab in the starting materials.
  3. Open GroupDetailPage.xaml and add the following statements after the <Page.Resources> section:
     1. XAML
     2. <Page.BottomAppBar>
     3. <AppBar x:Name="PageAppBar" Padding="10,0,10,0">
     4. <Grid>
     5. <Grid.ColumnDefinitions>
     6. <ColumnDefinition Width="50\*"/>
     7. <ColumnDefinition Width="50\*"/>
     8. </Grid.ColumnDefinitions>
     9. <StackPanel x:Name="LeftCommands" Orientation="Horizontal" Grid.Column="0" HorizontalAlignment="Left">
     10. </StackPanel>
     11. <StackPanel x:Name="RightCommands" Orientation="Horizontal" Grid.Column="1" HorizontalAlignment="Right">
     12. <Button x:Name="Home" HorizontalAlignment="Right" Style="{StaticResource HomeAppBarButtonStyle}" Click="OnHomeButtonClicked" />
     13. </StackPanel>
     14. </Grid>
     15. </AppBar>
     16. </Page.BottomAppBar>
     17. **Note:** The button, or “command,” declared in this application bar gets its look from the HomeAppBarButtonStyle style that Visual Studio built into StandardStyles.xaml, which is located in the project’s Common folder. StandardStyles.xaml contains styles for about 30 different types of application bar commands. If you examine these styles, you’ll find that the symbols on the face of the commands are specified using hexadecimal values such as xE10F, which are character codes in the Segoe UI Symbol character set. To find cool icons for your own commands, fire up Windows 8’s Character Map application, select Segoe UI Symbol in the drop-down font list, and scroll to the bottom. Instant iconography!

****

* 1. Open GroupDetailPage.xaml.cs and add the following handler for application bar command:
     1. C#
     2. private void OnHomeButtonClicked(object sender, RoutedEventArgs e)
     3. {
     4. // Navigate to the start page
     5. this.Frame.Navigate(typeof(GroupedItemsPage), "AllGroups");
     6. }
  2. Press F5 to run the application, and tap a group name to go to the group-detail page.
  3. Display the application bar by swiping upward from the bottom of the screen or pressing Win-Z.
  4. Verify that the right end of the application bar contains a Home command, as shown in Figure 1.
  5. 
  6. Figure 1
  7. The group-detail page’s application bar
  8. Tap the Home command and verify that you return to the app’s start page.
  9. Return to Visual Studio and stop debugging.

Task 2 – Add an Application Bar to the Item-Detail Page

Now let’s add an application bar to the item-detail page. This time, we’ll include commands for capturing photos and videos.

* 1. Open ItemDetailPage.xaml and add the following statements after the <Page.Resources> section:
     1. XAML
     2. <Page.BottomAppBar>
     3. <AppBar x:Name="PageAppBar" Padding="10,0,10,0">
     4. <Grid>
     5. <Grid.ColumnDefinitions>
     6. <ColumnDefinition Width="50\*"/>
     7. <ColumnDefinition Width="50\*"/>
     8. </Grid.ColumnDefinitions>
     9. <StackPanel x:Name="LeftCommands" Orientation="Horizontal" Grid.Column="0" HorizontalAlignment="Left">
     10. <Button x:Name="Photo" HorizontalAlignment="Left" Style="{StaticResource PhotoAppBarButtonStyle}" Click="OnPhotoButtonClicked" />
     11. <Button x:Name="Video" HorizontalAlignment="Left" Style="{StaticResource VideoAppBarButtonStyle}" Click="OnVideoButtonClicked" />
     12. </StackPanel>
     13. <StackPanel x:Name="RightCommands" Orientation="Horizontal" Grid.Column="1" HorizontalAlignment="Right">
     14. <Button x:Name="Home" HorizontalAlignment="Right" Style="{StaticResource HomeAppBarButtonStyle}" Click="OnHomeButtonClicked" />
     15. </StackPanel>
     16. </Grid>
     17. </AppBar>
     18. </Page.BottomAppBar>
  2. Open ItemDetailPage.xaml.cs and add the following methods:
     1. C#
     2. private void OnPhotoButtonClicked(object sender, RoutedEventArgs e)
     3. {
     4. DataTransferManager.ShowShareUI();
     5. }
     6. private void OnVideoButtonClicked(object sender, RoutedEventArgs e)
     7. {
     8. DataTransferManager.ShowShareUI();
     9. }
     10. private void OnHomeButtonClicked(object sender, RoutedEventArgs e)
     11. {
     12. // Navigate to the start page
     13. this.Frame.Navigate(typeof(GroupedItemsPage), "AllGroups");
     14. }
     15. **Note:** The DataTransfer.DataTransferManager.ShowShareUI method displays the share UI, which is the same UI that appears when you select the Share charm in the charms bar. You’re stubbing out the handlers for the Photo and Video commands here to show the share UI.
  3. Press F5 to run the application, and tap a recipe to go to the item-detail page.
  4. Display the application bar by swiping upward from the bottom of the screen or pressing Win-Z.
  5. Verify that the application bar contains a Photo command, a Video command, and a Home command, as shown in Figure 2.
  6. 
  7. Figure 2
  8. The item-detail page’s application bar
  9. Tap the Photo command and verify that Metro’s Sharing pane appears.
  10. Dismiss the sharing page. Then tap the Home command and verify that you return to the app’s start page.
  11. Return to Visual Studio and stop debugging.

Exercise 2: Add Photo Capture

1. The UI for snapping a photo is in place. Now let’s modify the code to allow the user to take a photo and share it with other applications. The Windows Runtime’s Windows.Media.Capture namespace includes a class named CameraCaptureUI class that provides a high-level interface to camera hardware and makes interfacing with cameras about as simple as it could possibly be.

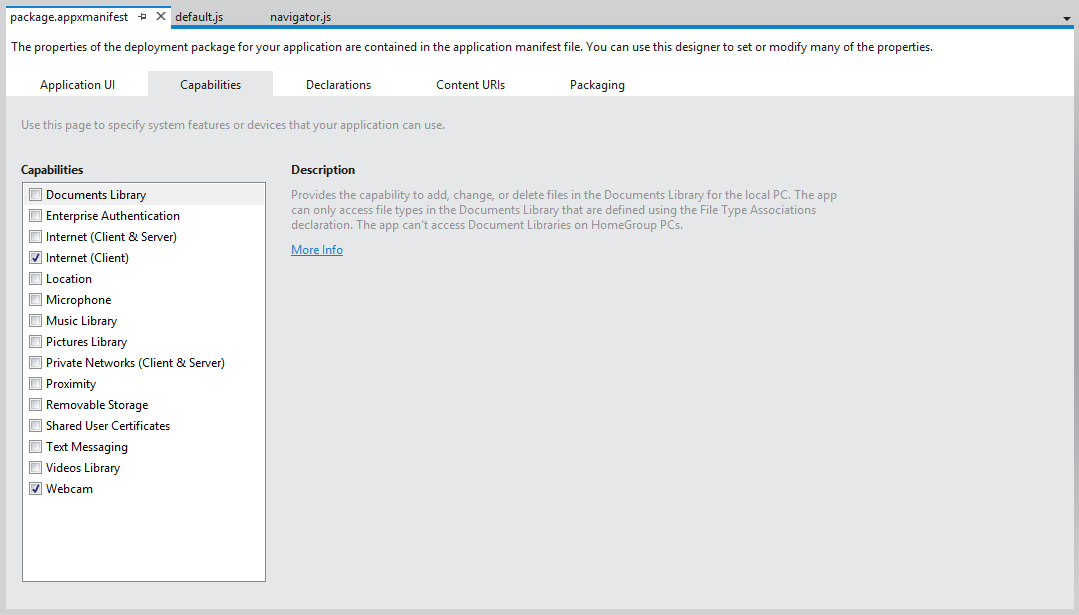
Task 1 – Use CameraCaptureUI to Capture Photos

CameraCaptureUI has a CaptureFileAsync method that makes short work of snapping photos, so let’s put it to work in Contoso Cookbook. While you’re at it, let’s modify the sharing code in the item-detail page so that it can share captured photos as well as recipe images.

* 1. Open ItemDetailPage.xaml.cs and add the following using statements at the top of the file:
     1. C#
     2. using Windows.Media.Capture;
     3. using Windows.Storage;
  2. Add the following field to the ItemDetailPage class:
     1. C#
     2. private StorageFile \_file; // Photo or video file to share
  3. Find the OnPhotoButtonClicked method you added in the previous exercise.
  4. Modify the method to look like this. **Be sure to add the async keyword on the first line**. It’s required since we’re using C#’s await keyword in the method body:
     1. C#
     2. private async void OnPhotoButtonClicked(object sender, RoutedEventArgs e)
     3. {
     4. var camera = new CameraCaptureUI();
     5. var file = await camera.CaptureFileAsync(CameraCaptureUIMode.Photo);
     6. if (file != null)
     7. {
     8. \_file = file;
     9. DataTransferManager.ShowShareUI();
     10. }
     11. }
  5. Find the OnDataRequested method you added to ItemDetailPage.xaml.cs in the previous lab.
  6. Apply the highlighted changes to the method. These changes are needed to share a recipe if \_file is null, or share a photo (or video) if it’s not null:
     1. C#
     2. void OnDataRequested(DataTransferManager sender, DataRequestedEventArgs args)
     3. {
     4. var request = args.Request;
     5. request.Data.Properties.Title = \_item.Title;
     6. if (\_file == null)
     7. {
     8. request.Data.Properties.Description = "Recipe ingredients and directions";
     9. // Share recipe text
     10. StringBuilder builder = new StringBuilder();
     11. builder.Append("INGREDIENTS\r\n");
     12. foreach (string ingredient in \_item.Ingredients)
     13. {
     14. builder.Append(ingredient);
     15. builder.Append("\r\n");
     16. }
     17. builder.Append("\r\nDIRECTIONS\r\n");
     18. builder.Append(\_item.Directions);
     19. request.Data.SetText(builder.ToString());
     20. // Share recipe image
     21. string url = \_item.GetImageUri();
     22. if (!url.StartsWith("http://"))
     23. url = "ms-appx:///" + url;
     24. var uri = new Uri(url);
     25. var reference = RandomAccessStreamReference.CreateFromUri(uri);
     26. request.Data.Properties.Thumbnail = reference;
     27. var deferral = request.GetDeferral();
     28. request.Data.SetBitmap(reference);
     29. deferral.Complete();
     30. }
     31. else
     32. {
     33. request.Data.Properties.Description = "Recipe photo or video";
     34. var reference = Windows.Storage.Streams.RandomAccessStreamReference.CreateFromFile(\_file);
     35. request.Data.Properties.Thumbnail = reference;
     36. request.Data.SetBitmap(reference);
     37. \_file = null;
     38. }
     39. }
  7. Press F5 to run the application and tap a recipe to go to the item-detail page.
  8. Display the application bar and tap the Photo button. What happens?
  9. Return to Visual Studio and stop debugging.

Task 2 – Enable Webcam Access

Metro-style apps must have permission to access webcams. That permission comes through the application manifest, which contains metadata about a Metro-style app. The next step, then, is to edit Contoso Cookbook’s manifest to indicate that it requires access to webcams.

* 1. In Solution Explorer, double-click Package.appxmanifest to open it for editing.
  2. Go to the Capabilities tab and check the Webcam box, as shown in Figure 3.
  3. 
  4. Figure 3
  5. Enabling webcam access in the application manifest

Taks 3 – Test the Results

1. Now that webcam access is enabled, let’s capture a photo.
   1. Press F5 to run the application and tap a recipe to go to the item-detail page.
   2. Display the application bar and tap the Photo button. Click “Allow” if asked if the app can use your webcam.
   3. When the camera-capture UI appears, tap the screen to snap a photo.
   4. Tap the OK button in the lower-right corner of the screen to accept the photo.
   5. When the share UI appears, select a share target such as Share Target Sample App.
   6. Confirm that the photo you just captured is accepted by the share target.
   7. Return to Visual Studio and stop debugging.

Exercise 3: Add Video Capture

1. Contoso Cookbook users can now snap photos and share them with other applications. In this exercise, you’ll add support for capturing videos, too. You’ll change the parameter passed to CaptureFileAsync to indicate that you want to capture video rather than photos, and use CameraCaptureUI’s VideoSettings property to indicate what format you want to capture in.

Task 1 – Use CameraCaptureUI to Capture Video

The same CaptureFileAsync method that captures a photo can be used to capture video, too. Let’s modify the OnVideoButtonClicked method you stubbed out earlier to demonstrate.

* 1. Find the OnVideoButtonClicked method you added to ItemDetailPage.xaml.cs in Exercise 1.
  2. Modify the method to look like this. Once more, be sure to include the async keyword on the first line. It’s required since we’re using C#’s await keyword in the method body:
     1. C#
     2. private async void OnVideoButtonClicked(object sender, RoutedEventArgs e)
     3. {
     4. var camera = new CameraCaptureUI();
     5. var file = await camera.CaptureFileAsync(CameraCaptureUIMode.Video);
     6. camera.VideoSettings.Format = CameraCaptureUIVideoFormat.Wmv;
     7. if (file != null)
     8. {
     9. \_file = file;
     10. DataTransferManager.ShowShareUI();
     11. }
     12. }
  3. Take a moment to examine the code you just added. How does it differ from the code for the OnPhotoButtonClicked method? What is it that tells CameraCaptureUI to capture a video rather than a photo?
  4. Go to the Capabilities section of the app manifest and check the “Microphone” box. This is necessary because when you capture video, CameraCaptureUI uses the microphone as well as the camera.

Task 2 – Test the Results

Now let’s test the code you just added.

* 1. Press F5 to run the application and tap a recipe to go to the item-detail page.
  2. Display the application bar and tap the Video button.
  3. If asked whether the app can use your camera and microphone, click “Allow.”
  4. When the camera-capture UI appears, tap the screen to begin capturing video.
  5. After a few seconds, tap the screen again to stop capturing video.
  6. Tap the OK button in the lower-right corner of the screen to accept the video. At this point, you’re done, because there is probably no share target on your device that accepts videos.
  7. Return to Visual Studio and stop debugging.

Summary

* 1. On some platforms, including photo and video capture capabilities in an application is a difficult undertaking, requiring you to interface with cameras at the device level. The Windows Runtime makes media capture extraordinarily easy by providing the core UI and logic in the CameraCaptureUI class. In Contoso Cookbook, we don’t do anything with captured image and video files other than share them out. However, you could easily use types in the Windows.Storage namespace to save these files to the file system and allow the user to create libraries of recipe photos and videos.
  2. You may not have noticed, but the operating system did something pretty cool for you when checked the Webcam and Microphone boxes in the manifest. If you run the app, select the Settings charm, and select “Permissions,” you’ll see that the permissions page now contains a toggle button for turning camera and microphone access on and off. Apps that use cameras and microphones are required to allow access to be disabled by the user, and you didn’t have to write a single line of code to make it happen.
  3. Speaking of the Settings charm: How would like to be able to customize the settings pane by adding an about page, a preferences page, and perhaps other pages as well? Funny you should ask, because that’s what you’re going to do in Lab 6. But first, we have a more pressing topic to talk about: Process Lifetime Management, or PLM.