Hands-On Lab

Settings and Preferences

Lab version: 1.0.0

Last updated: 6/20/2012



Contents

[Overview 3](#_Toc327916585)

[Exercise 1: Add an About Page 4](#_Toc327916586)

[Task 1 – Add an About Command 4](#_Toc327916587)

[Task 2 – Add an about.html Page Control 5](#_Toc327916588)

[Task 3 – Test the Results 7](#_Toc327916589)

[Exercise 2: Add a Preferences Page 8](#_Toc327916590)

[Task 1 – Add a Preferences Command 8](#_Toc327916591)

[Task 2 – Add a preferences.html Page Control 9](#_Toc327916592)

[Task 3 – Make the Preference Sticky 10](#_Toc327916593)

[Exercise 3: Implement the Preference 12](#_Toc327916594)

[Task 1 – Modify default.js 12](#_Toc327916595)

[Task 2 – Test the Results 13](#_Toc327916596)

[Summary 14](#_Toc327916597)

Overview

* 1. Lab 3 introduced Metro’s charms bar and demonstrated how easily applications can integrate with the Share and Search charms. The charms bar also includes a Settings charm, which is used to change settings in the active Metro application. In the settings pane that appears when you select the Settings charm, the operating system provides a Permissions command that allows users to enable and disable certain features of the program such as webcam and microphone access. Significantly, you can add commands of your own to the settings pane and connect them to HTML page controls, providing users with convenient access to preferences, about boxes, and other application-specific settings content.
  2. In this lab, you’ll add About and Preferences commands to the settings pane in Contoso Cookbook. You’ll expose a simple user preference that can be toggled on and off with a checkbox, and you’ll use roaming settings to store that preference so it will follow users wherever they go.

# Objectives

* 1. This lab will show you how to:
  + Add an About command and an about page to the settings pane
  + Add a Preferences command and a preferences page to the settings pane
  + Use roaming settings to store user preferences

# 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 the Microsoft Windows 8 Release Preview
  3. Install the Microsoft Visual Studio 2012 RC

# Exercises

* 1. This Hands-On Lab comprises the following exercises:
  2. Add an About Page
  3. Add a Preferences Page
  4. Implement the Preference
  5. Estimated time to complete this lab: **30 to 40 minutes**.

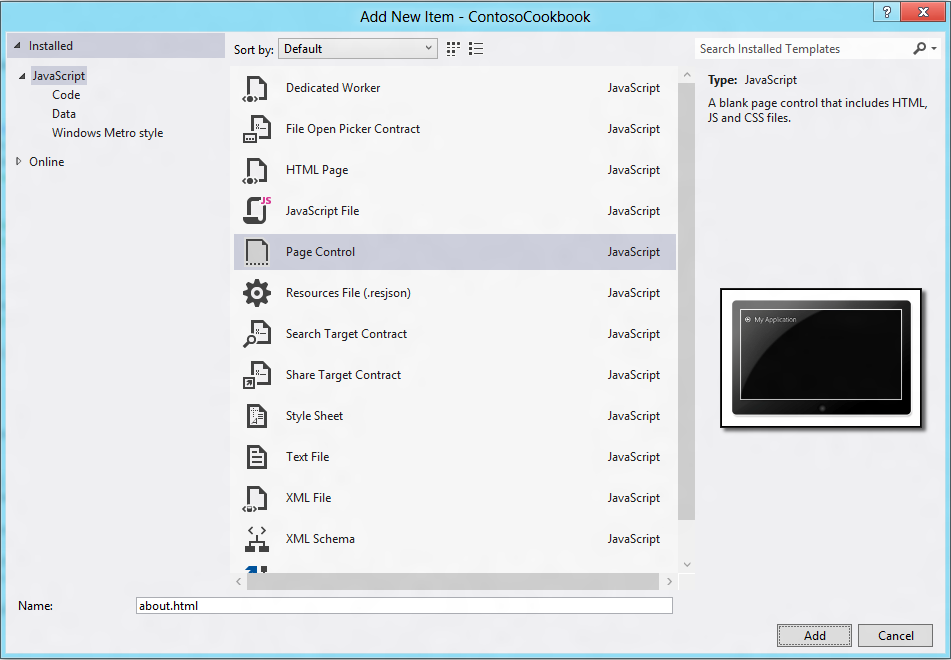
Exercise 1: Add an About Page

1. In this exercise, you’ll add a simple about page to Contoso Cookbook.

Task 1 – Add an About Command

* 1. The first step in adding commands to Metro’s settings pane is to register a handler for the WinJS.Application.onsettings event. This event fires whenever the settings pane is displayed, and an event handler can add commands to the settings menu.
  2. Open the ContosoCookbook project you finished in Lab 5 in Visual Studio. If you didn’t complete Lab 5 or would like to start with a reference copy, you’ll find a completed version of the lab in the starting materials.
  3. Open default.js and add the following statements before the if (app.sessionState.history) statement:
     1. JavaScript
     2. app.onsettings = function (e) {
     3. e.detail.applicationcommands = {
     4. // Add an About command
     5. "about": {
     6. href: "/pages/about/about.html",
     7. title: "About"
     8. }
     9. }
     10. WinJS.UI.SettingsFlyout.populateSettings(e);
     11. };
  4. Press F5 to start the application.
  5. Display the charms bar and select Settings.
  6. Verify that the settings pane contains an About command, as shown in Figure 1.
     1. 
     2. Figure 1
     3. The settings pane with an About command
  7. Return to Visual Studio and stop debugging.

Task 2 – Add an about.html Page Control

* 1. The event handler you added in the previous task adds an About command to the settings pane and points it to /pages/about/about.html, but as of right now, that page doesn’t exist. Let’s fix that by adding a new page control to the project.
  2. In Solution Explorer, right-click the pages folder and use the **Add - New Folder** command to add a folder named about.
  3. Right-click the about folder and use the **Add - New Item** command to add a page control named about.html, as shown in Figure 2.
     1. 
     2. Figure 2
     3. Adding a page control
  4. Open about.html and replace the DIV whose ID is “about fragment” with this one:
     1. HTML
     2. <div id="about" data-win-control="WinJS.UI.SettingsFlyout" data-win-options="{width: 'narrow'}">
     3. <div class="SettingsPane">
     4. <div class="win-label">
     5. <button onclick="WinJS.UI.SettingsFlyout.show()" class="win-backbutton">
     6. </button>
     7. <span class="SettingsTitle">About</span>
     8. </div>
     9. <article class="SettingsContent">
     10. <h2>Contoso Cookbook</h2>
     11. <h4>Trial Version</h4>
     12. </article>
     13. </div>
     14. </div>
     15. **Note:** The id=”about” attribute on the DIV is critical because it maps back to the “about” you passed to e.detail.applicationcommands in the app.onsettings handler in Task 1. The about page currently informs the user that this is a trial version of Contoso Cookbook. In Lab 7, you’ll use the Windows Runtime’s store APIs to simulate purchases of the app, and once a purchase occurs, you’ll replace “Trial Version” with licensing information.
  5. Open about.css and add the following CSS classes:
     1. CSS
     2. .SettingsPane {
     3. margin-top:36px;
     4. margin-left:48px;
     5. }
     6. .SettingsTitle {
     7. margin-left: 36px;
     8. }
     9. .SettingsContent {
     10. margin-top: 24px;
     11. }

Task 3 – Test the Results

* 1. Now it’s time to test your changes and see what a Metro-style about page looks like.
  2. Press F5 to run the application.
  3. Display the charms bar and select Settings.
  4. Select the About command from the settings pane.
  5. Confirm that the about page appears, as shown in Figure 3.
     1. 
     2. Figure 3
     3. Contoso Cookbook’s about page
  6. Return to Visual Studio and stop debugging.

Exercise 2: Add a Preferences Page

1. Now that you understand the mechanics of adding a page to Metro’s settings pane, you’ll add another page – this time, a preferences page that allows the user to enter and edit preferences. We’ll just add one preference to demonstrate how it’s done, but of course you’re free to add as many preferences as you’d like. The preference you’ll add is one that allows the user to configure Contoso Cookbook to return to the last recipe or recipe group that was displayed when it starts up.

Task 1 – Add a Preferences Command

* 1. Start by modifying the onsettings handler that adds an About command to the settings pane so that it adds a Preferences command, too.
  2. Open default.js and find the app.onsettings function you added in the previous exercise.
  3. Modify the function to include a Preferences command:
     1. JavaScript
     2. app.onsettings = function (e) {
     3. e.detail.applicationcommands = {
     4. // Add an About command
     5. "about": {
     6. href: "/pages/about/about.html",
     7. title: "About"
     8. },
     9. "preferences": {
     10. href: "/pages/preferences/preferences.html",
     11. title: "Preferences"
     12. }
     13. }
     14. WinJS.UI.SettingsFlyout.populateSettings(e);
     15. };
  4. Press F5 to start the application, and use the Settings charm to display the settings pane.
  5. Confirm that the settings pane now includes a Preferences command, as shown in Figure 4.
     1. 
     2. Figure 4
     3. The settings pane with a Preferences command
  6. Return to Visual Studio and stop debugging.

Task 2 – Add a preferences.html Page Control

* 1. The next task is to create the page control that the Preferences command invokes.
  2. In Solution Explorer, right-click the pages folder and use the **Add - New Folder** command to add a folder named preferences.
  3. Right-click the preferences folder and use the **Add - New Item** command to add a page control named preferences.html.
  4. Open preferences.html and replace the DIV whose ID is “preferences fragment” with this one:
     1. HTML
     2. <div id="preferences" data-win-control="WinJS.UI.SettingsFlyout" data-win-options="{width: 'narrow'}">
     3. <div class="SettingsPane">
     4. <div class="win-label">
     5. <button onclick="WinJS.UI.SettingsFlyout.show()" class="win-backbutton">
     6. </button>
     7. <span class="SettingsTitle">Preferences</span>
     8. </div>
     9. <article class="SettingsContent">
     10. <div id="remember" data-win-control="WinJS.UI.ToggleSwitch" data-win-options="{ title: 'Remember where I was' }" />
     11. </article>
     12. </div>
     13. </div>
  5. Open preferences.css and add the following CSS classes:
     1. CSS
     2. .SettingsPane {
     3. margin-top:36px;
     4. margin-left:48px;
     5. }
     6. .SettingsTitle {
     7. margin-left: 36px;
     8. }
     9. .SettingsContent {
     10. margin-top: 24px;
     11. }
  6. Press F5 to run the application.
  7. Display the charms bar and select Settings.
  8. Select the Preferences command from the settings pane.
  9. Confirm that the preferences page appears and that it contains a toggle switch, as shown in Figure 5.
     1. 
     2. Figure 5
     3. Contoso Cookbook’s preferences page
  10. Return to Visual Studio and stop debugging.

Task 3 – Make the Preference Sticky

* 1. Right now, the “Remember where I was” toggle switch in the preferences page isn’t wired up to anything, and it doesn’t retain its state. Let’s fix that by using roaming settings to save the state of the toggle switch each time it’s clicked, and to initialize it each time the preferences page is displayed.
  2. Open preferences.js and add the following statement after the “use strict” line:
     1. JavaScript
     2. var appdata = Windows.Storage.ApplicationData;
  3. Also in preferences.js, add the following statements to the ready function to record the state of the toggle switch in roaming settings each time it’s clicked, and to initialize the toggle switch each time the preferences page is displayed:
     1. JavaScript
     2. var toggle = document.querySelector("#remember").winControl;
     3. var remember = appdata.current.roamingSettings.values["remember"];
     4. remember = !remember ? false : remember; // false if value doesn’t exist
     5. toggle.checked = remember;
     6. toggle.addEventListener("change", function (e) {
     7. appdata.current.roamingSettings.values["remember"] = e.target.winControl.checked;
     8. });
     9. **Note:** To assist with the task of saving and restoring settings and other application data, the Windows Runtime gives you the Windows.Storage.ApplicationData class. ApplicationData lets you store data locally, in the cloud (roaming storage), or in temporary storage. The data you save can be stored in the form of name-value pairs through ApplicationData properties named LocalSettings and RoamingSettings, or it can be stored in files created in special app-specific folders accessed through ApplicationData’s LocalFolder, RoamingFolder, and TemporaryFolder properties.
     10. The benefit of persisting data in one of the roaming repositories is that such data “follows” the user from one device to another. Moreover, if an application writes data to RoamingSettings or RoamingFolder and the user isn’t logged in with a Microsoft account or doesn’t have an Internet connection, the Windows Runtime automatically persists the data locally. So there’s little to lose and lots to gain by using RoamingSettings or RoamingFolder as a store for user preferences. The only caveat is that the platform limits how much data can be saved in roaming storage. In the Release Preview, you can roam about 100K of data. You can determine at run-time how much quota you have from the ApplicationData.RoamingStorageQuota property.
  4. Press F5 to run the application.
  5. Display the charms bar and select Settings.
  6. Select the Preferences command from the settings pane.
  7. Tap “Remember where I was” to turn the toggle switch on.
  8. Dismiss the settings pane.
  9. Return to Visual Studio and stop debugging.
  10. Press F5 to start the application again.
  11. Go to the preferences page and confirm that the toggle switch is on.
  12. Return to Visual Studio and stop debugging.

Exercise 3: Implement the Preference

1. Currently, Contoso Cookbook shows the start page each time it starts up. The purpose of adding a user preference entitled “Remember where I was” in the previous exercise was to allow a user to configure the application to return each time it starts up to the page that was displayed the last time it shut down. Honoring this user preference will require only minor changes to your code since Visual Studio has already included code in your app that saves the navigation state when the app is suspended.

Task 1 – Modify default.js

* 1. default.js already contains code to save the navigation state when the app is suspended and restore it if it’s terminated. We’ll use a similar strategy to restore the navigation state if the app starts up after being closed by the user and “Remember where I was” is enabled.
  2. Open default.js and add the following statement after the statement that defines a variable named nav near the top of the file:
     1. JavaScript
     2. var appdata = Windows.Storage.ApplicationData;
  3. Add the following statements to the activated event handler, just after the statement that reads “if (args.detail.kind === activation.ActivationKind.launch)”:
     1. JavaScript
     2. // If "Remember where I was" is enabled and roaming settings
     3. // contains a history, apply it to go back to where the user
     4. // was before
     5. var remember = appdata.current.roamingSettings.values["remember"];
     6. remember = !remember ? false : remember; // false if value is undefined
     7. if (remember) {
     8. var history = appdata.current.roamingSettings.values["history"];
     9. if (history !== undefined) {
     10. nav.history = JSON.parse(history);
     11. }
     12. }
  4. Add the following statements to the end of the app.oncheckpoint handler near the bottom of default.js:
     1. JavaScript
     2. // If "Remember where I was" is enabled, write the history to
     3. // roaming settings so it can be restored later
     4. var remember = appdata.current.roamingSettings.values["remember"];
     5. remember = !remember ? false : remember; // false if value is undefined
     6. if (remember) {
     7. appdata.current.roamingSettings.values["history"] = JSON.stringify(nav.history);
     8. }

Task 2 – Test the Results

* 1. All that remains is to do a little testing to ensure that the change works.
  2. Press F5 to run the application.
  3. Display the charms bar and select Settings.
  4. Select the Preferences command from the settings pane.
  5. Verify that “Remember where I was” is still on. If it’s not, turn it on.
  6. Dismiss the settings pane.
  7. Navigate to a recipe page.
  8. Close the application by swiping downward from the top of the screen or pressing Alt+F4. (*Do not* close the app with Visual Studio’s **Stop Debugging** command.)
  9. Return to Visual Studio and wait a few seconds for the process to end. (It generally takes about 10 seconds.)
  10. Press F5 to launch the application again.
  11. Confirm that Contoso Cookbook returns you to the recipe displayed when you closed it.
  12. Go to the preferences page and toggle “Remember where I was” off.
  13. Close the application again by swiping downward from the top of the screen or pressing Alt+F4 while viewing a recipe page.
  14. Return to Visual Studio and wait for the process to end.
  15. Press F5 to launch the application again.
  16. Confirm that you go to the start page, and *not* to the recipe you last viewed.
  17. Return to Visual Studio and stop debugging.

Summary

* 1. Settings and preferences are an important part of virtually every Metro application. Metro’s Settings charm provides a familiar and consistent model for viewing and editing application settings, and as you learned in this lab, it’s simple for a Metro-style app app to expose content through the settings pane.
  2. We’ve come a long way since Lab 1, but there is still more to do. Next up is another important step on the road to Metro stardom: tiles and notifications.