Hands-On Lab

The Windows Store

Lab version: 1.0.0

Last updated: 4/1/2012



Contents

[Overview 3](#_Toc320972599)

[Exercise 1: Detect Trial Versions 4](#_Toc320972600)

[Task 1 – Add WindowsStoreProxy.xml 4](#_Toc320972601)

[Task 2 – Modify the About Page 5](#_Toc320972602)

[Task 3 – Test the Results 8](#_Toc320972603)

[Exercise 2: Simulate App Purchases 10](#_Toc320972604)

[Task 1 – Write a Handler for the Purchase Button 10](#_Toc320972605)

[Task 2 – Purchase the App 11](#_Toc320972606)

[Exercise 3: Simulate Product Purchases 12](#_Toc320972607)

[Task 1 – Modify the Item-Detail Page 12](#_Toc320972608)

[Task 2 – Make a Product Purchase 15](#_Toc320972609)

[Summary 16](#_Toc320972610)

Overview

One of the most compelling reasons to write Metro applications is the ease with which you can publish them in the Windows Store. With more than 500 million PCs worldwide currently running Windows 7, and with each of those PCs representing a potential upgrade to Windows 8, the market – and revenue potential – is both huge and diverse. And with a revenue sharing plan that directs up to 80% of the sales proceeds to the author, there’s no shortage of motivation for developers to write great applications and offer them to the world.

You can make apps published through the Windows Store available for free, or you can monetize your efforts by charging for them. You can also choose to make trial versions available and use the Windows Store APIs in Windows.ApplicationModel.Store to detect trial versions and, if you choose, offer reduced functionality to the user until the app is paid for. Other APIs make it easy to upgrade from trial versions to paid versions, support in-app purchases of additional products, retrieve licensing information, and more. And the Windows Runtime’s CurrentAppSimulator class provides a handy means for simulating purchases and testing code that relies on Windows Store APIs in a controlled environment.

In this lab, you’ll use the Windows Store APIs to monetize Contoso Cookbook. First you’ll modify the about box to detect trial versions and include a purchase button if the app hasn’t been paid for. Next you’ll use CurrentAppSimulator to simulate a purchase when the purchase button is clicked. Finally, you’ll simulate in-app purchases by offering Italian recipes as a paid add-on rather than for free.

# Objectives

This lab will show you how to:

* + Detect whether your app is running as a trial version
  + Simulate purchases of the app from within the app itself
  + Simulate in-app purchases of additional products
  + Retrieve licensing information regarding apps and products

# 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

This Hands-On Lab comprises the following exercises:

* 1. Detect Trial Versions
  2. Simulate App Purchases
  3. Simulate Product Purchases

Estimated time to complete this lab: **30 to 40 minutes**.

Exercise 1: Detect Trial Versions

1. In this exercise, you’ll use the Windows Runtime’s store APIs to customize the content of Contoso Cookbook’s about page. If the app has been purchased, you’ll display licensing information. If it hasn’t been purchased – that is, if it’s running as a trial version – then you’ll display a purchase button instead. Moreover, the price displayed on the purchase button won’t be hardcoded, but will come from listing information retrieved from the Windows Store.

Task 1 – Add WindowsStoreProxy.xml

We’ll be using the CurrentAppSimulator class to simulate purchases, retrieve licensing information, and more. To make the simulation as realistic as possible, we’ll use a file named WindowsStoreProxy.xml to provide information about pricing, expiration dates, and more to CurrentAppSimulator.

* 1. Open the ContosoCookbook project you finished in Lab 7 in Visual Studio. If you didn’t complete Lab 7 or would like to start with a reference copy, you’ll find a completed version of the lab in the starting materials.
  2. If the project doesn’t already have a Data folder, create one.
  3. Right-click the Data folder and use the **Add - Existing Item** command to import license.xml from the data folder of the starting materials. This is the file from which we’ll create WindowsStoreProxy.xml.
  4. Open App.xaml.cs and add the following statements to the OnLaunched method. Position this code *before* the statement that calls CreatePushNotificationChannelForApplicationAsync that you added in the previous lab:
     1. C#
     2. // Initialize WindowsStoreProxy.xml
     3. var folder = await ApplicationData.Current.LocalFolder.CreateFolderAsync("Microsoft\\Windows Store\\ApiData", CreationCollisionOption.OpenIfExists);
     4. var file = await Package.Current.InstalledLocation.GetFileAsync("Data\\license.xml");
     5. var newfile = await folder.CreateFileAsync("WindowsStoreProxy.xml", CreationCollisionOption.ReplaceExisting);

await file.CopyAndReplaceAsync(newfile);

* + 1. **Note:** The code you just added use the Windows Runtime’s storage API to create WindowsStoreProxy.xml in the specific location in which CurrentAppSimulator expects to find it and copies the contents of license.xml into it each time the app starts up. You can simulate app purchases and product purchases without a proxy file, but if you want a richer simulation that retrieves prices and other information, you must include WindowsStoreProxy.xml.
  1. Open license.xml and take a moment to inspect its contents. The <ListingInformation> element contains information about the app itself and about the Italian-recipes product we’ll offer for purchase in Exercise 3. <LicenseInformation> contains licensing information about the app and the product. In real life, all of this information would come from the Windows Store. In a simulation, however, the information comes from WindowsStoreProxy.xml.

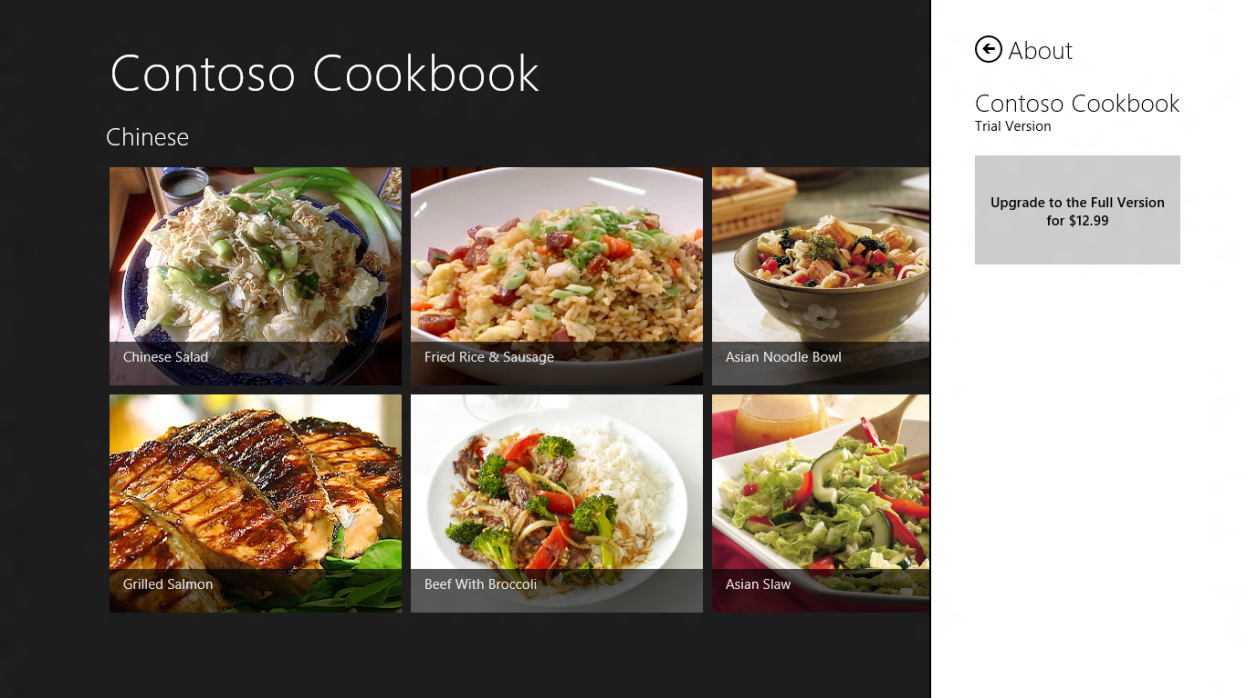
Task 2 – Modify the About Page

Now let’s modify the about page you created in Lab 6. Currently, the words “Trial Version” appear underneath the app title in the about page. We’ll use store APIs to determine whether this is indeed a trial version and customize the page’s content based on the results.

* 1. Right-click the project’s DataModel folder and use the **Add - New Item** command to add a new class to the project. Name the file AppLicenseDataSource.cs.
  2. Replace the file’s contents with this:
     1. C#
     2. using System;
     3. using System.Collections.Generic;
     4. using System.ComponentModel;
     5. using System.Linq;
     6. using System.Text;
     7. using System.Threading.Tasks;
     8. using Windows.ApplicationModel.Store;
     9. using Windows.Foundation;
     10. namespace ContosoCookbook
     11. {
     12. class AppLicenseDataSource : INotifyPropertyChanged
     13. {
     14. public event PropertyChangedEventHandler PropertyChanged;
     15. private bool \_licensed = false;
     16. private string \_price;
     17. public AppLicenseDataSource()
     18. {
     19. if (CurrentAppSimulator.LicenseInformation.IsTrial)
     20. {
     21. CurrentAppSimulator.LicenseInformation.LicenseChanged += OnLicenseChanged;
     22. GetListingInformationAsync();
     23. }
     24. else
     25. \_licensed = true;
     26. }
     27. private async void GetListingInformationAsync()
     28. {
     29. var listing = await CurrentAppSimulator.LoadListingInformationAsync();
     30. \_price = listing.FormattedPrice;
     31. }
     33. private void OnLicenseChanged()
     34. {
     35. if (!CurrentAppSimulator.LicenseInformation.IsTrial)
     36. {
     37. \_licensed = true;
     38. CurrentAppSimulator.LicenseInformation.LicenseChanged -= OnLicenseChanged;
     39. if (PropertyChanged != null)
     40. {
     41. PropertyChanged(this, new PropertyChangedEventArgs("IsLicensed"));
     42. PropertyChanged(this, new PropertyChangedEventArgs("IsTrial"));
     43. PropertyChanged(this, new PropertyChangedEventArgs("LicenseInfo"));
     44. }
     45. }
     46. }
     47. public bool IsLicensed
     48. {
     49. get { return \_licensed; }
     50. }
     51. public bool IsTrial
     52. {
     53. get { return !\_licensed; }
     54. }
     55. public string LicenseInfo
     56. {
     57. get
     58. {
     59. if (!\_licensed)
     60. return "Trial Version";
     61. else
     62. return ("Valid until " + CurrentAppSimulator.LicenseInformation.ExpirationDate.LocalDateTime.ToString("dddd, MMMM d, yyyy"));
     63. }
     64. }
     66. public string FormattedPrice
     67. {
     68. get
     69. {
     70. if (!String.IsNullOrEmpty(\_price))
     71. return "Upgrade to the Full Version for " + \_price;
     72. else
     73. return "Upgrade to the Full Version";
     74. }
     75. }
     76. }
     77. }
     78. **Note:** The AppLicenseDataSource class you just added implements INotifyPropertyChanged and exposes bindable properties named IsLicensed, IsTrial, LicenseInfo, and FormattedPrice. The first two use CurrentAppSimulator.LicenseInformation.IsTrial to determine whether the app that’s currently running has been purchased or is running as a trial version. LicenseInfo returns the string “Trial Version” if the app is a trial or a string containing the license’s expiration date if it’s not. FormattedPrice returns a button label containing the app’s price, which is obtained from the Windows Store (or, in this case, WindowsStoreProxy.xml). Pricing information comes from the ListingInformation object retrieved with CurrentAppSimulator.LoadListingInformationAsync.
  3. Open AboutUserControl.xaml and add the following attribute to the UserControl element at the top of the page:
     1. XAML
     2. xmlns:common="using:ContosoCookbook.Common"
  4. Now add the following statements just above the first Border element:
     1. XAML
     2. <UserControl.Resources>
     3. <local:AppLicenseDataSource x:Key="License" />
     4. <common:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>
     5. </UserControl.Resources>
     6. **Note:** BooleanToVisibilityConverter is a simple value converter that converts the Boolean value true into Visibility.Visible, and false into Visibility.Collapsed. It was included when the project was created by Visual Studio.
  5. Find the TextBlock whose text is “Trial Version” near the bottom of the page. Replace that TextBlock with the following statements:
     1. XAML
     2. <TextBlock Text="{Binding LicenseInfo, Source={StaticResource License}}" Foreground="Black" FontFamily="Segoe UI" FontWeight="SemiLight" FontSize="18" />
     3. <Button x:Name="PurchaseButton" Width="225" Height="120" Margin="0,24,0,0" Foreground="Black" Background="LightGray" BorderBrush="LightGray" Visibility="{Binding IsTrial, Source={StaticResource License}, Converter={StaticResource BooleanToVisibilityConverter }}">
     4. <Button.Content>
     5. <TextBlock Text="{Binding FormattedPrice, Source={StaticResource License}}" TextWrapping="Wrap" TextAlignment="Center" />
     6. </Button.Content>
     7. </Button>
  6. Take a moment to examine the XAML you just added. The TextBlock text now comes from the LicenseInfo property of the AppLicenseDataSource object. The button text comes from the FormattedPrice property of the same object, and the button’s visibility is linked to the IsTrial property. As a result, the button won’t even appear in the page unless the app is running as a trial version.

Task 3 – Test the Results

Now let’s test these changes to see CurrentAppSimulator and WindowsStoreProxy.xml in action.

* 1. Press F5 to launch the application.
  2. Display the charms bar and tap the Settings charm.
  3. Tap About in the settings menu to display the about page.
  4. Confirm that a purchase button appears in the about page, and that the purchase price is $12.99, as shown in Figure 1.
  5. 
  6. Figure 1
  7. The about page for the trial version of the app
  8. Return to Visual Studio and stop debugging.
  9. Open license.xml and change the purchase price in the <Price> element in the <App> section from $12.99 to $8.99.
  10. Launch the app again and go to the about page. What’s the price shown on the purchase button now?
  11. Return to Visual Studio once more and stop debugging.
  12. Open license.xml again and change the price back to $12.99. Also change <IsTrial> from true to false.
  13. Launch the app and go to the about page. Verify that the purchase button is gone and that you now see the “Valid until Saturday, December 31, 2022” message in Figure 2.
  14. 
  15. Figure 2
  16. The about page for the purchased version of the app
  17. Return to Visual Studio and stop debugging.
  18. In preparation for the next exercise, change <IsTrial> from false to true in license.xml.

Exercise 2: Simulate App Purchases

1. You can use WindowsStoreProxy.xml to test changes to your UI based on whether the app is a trial version, but that’s no substitute for being able to simulate actual purchases. In this exercise, you’ll write a handler for the purchase button so you can “purchase” the application from the Windows Store.

Task 1 – Write a Handler for the Purchase Button

To simulate app purchases, we’ll call CurrentAppSimulator.RequestAppPurchaseAsync when the user clicks the about page’s purchase button. And to detect when the purchase has completed successfully, we’ll use the LicenseChanged event handler already present in AppLicenseDataSource.

* 1. Open AboutUserControl.xaml.
  2. Add the highlighted Click attribute to the purchase button:
     1. XAML
     2. <Button x:Name="PurchaseButton" Width="225" Height="120" Margin="0,24,0,0" Foreground="Black" Background="LightGray" Visibility="{Binding IsTrial, Source={StaticResource License}, Converter={StaticResource BooleanToVisibilityConverter }}" Click="OnPurchaseButtonClicked">
     3. <Button.Content>
     4. <TextBlock Text="{Binding FormattedPrice, Source={StaticResource License}}" TextWrapping="Wrap" TextAlignment="Center" />
     5. </Button.Content>
     6. </Button>
  3. Open AboutUserControl.xaml.cs and add the following using statement:
     1. C#
     2. using Windows.ApplicationModel.Store;
  4. Then add the following event handler:
     1. C#
     2. private void OnPurchaseButtonClicked(object sender, RoutedEventArgs e)
     3. {
     4. // Purchase the app
     5. CurrentAppSimulator.RequestAppPurchaseAsync(true);
     6. }
     7. **Note:** RequestAppPurchaseAsync is an asynchronous method. To determine whether the purchase was carried out, you handle LicenseChanged events. You’re not handling those events here because you’re already handling them in AppLicenseDataSource.cs. When a LicenseChanged event fires, the handler verifies that the app is no longer a trial version and fires PropertyChanged events to update controls bound to IsTrial and other AppLicenseDataSource properties.

Task 2 – Purchase the App

Pretty simple, huh? Now let’s simulate an app purchase. Be aware that CurrentAppSimulator stores information regarding purchases and changes in licensing status in memory; it doesn’t record them in WindowsStoreProxy.xml. Therefore, once you purchase the app, it will remain “purchased” as long as the app is running, but when you restart the app, you’ll once more be running a trial version.

* 1. Press F5 to launch the application.
  2. Go to the about page and tap the purchase button to simulate an app purchase.
  3. Complete the simulated purchase by tapping the Continue button in the Windows Store dialog.
  4. Display the about page again and confirm that the purchase button has disappeared.
     1. **Note:** Contoso Cookbook doesn’t expose additional functionality to the user once it has been purchased; it merely replaces the purchase button with licensing information. In real life, you might choose to limit what the user can do with a trial version and only expose the full functionality once a purchase has been made.
  5. Return to Visual Studio and stop debugging.

Exercise 3: Simulate Product Purchases

1. In addition to allowing applications to be purchased, the Windows Store supports in-app product purchases, too. For example, a game program could allow users to purchase additional levels of the game as they complete previous levels. In Windows Store parlance, features purchased this way are known as *products*, and the Windows Runtime provides the APIs you need for product purchases, as well as APIs for determining which products have been purchased, the licensing status of those products, and more.
2. In this exercise, you’ll modify Contoso Cookbook so that Italian recipes are no longer free, but must be purchased. You’ll add a simple UI for purchasing them that relies on CurrentAppSimulator, and logic that prevents Italian recipes from being shown in full until after a product purchase has been made.

Task 1 – Modify the Item-Detail Page

The first step is to add a data-source class to provide product licensing information. Then we’ll add a purchase button and use data binding to make sure that for Italian recipes, either the purchase button or recipe directions are displayed, but not both.

* 1. Right-click the project’s DataModel folder and use the **Add - New Item** command to add a new class to the project. Name the file ProductLicenseDataSource.cs.
  2. Replace the file’s contents with this:
     1. C#
     2. using System;
     3. using System.Collections.Generic;
     4. using System.ComponentModel;
     5. using System.Linq;
     6. using System.Text;
     7. using System.Threading.Tasks;
     8. using Windows.ApplicationModel.Store;
     9. namespace ContosoCookbook
     10. {
     11. class ProductLicenseDataSource : INotifyPropertyChanged
     12. {
     13. public event PropertyChangedEventHandler PropertyChanged;
     14. private const string \_name = "ItalianRecipes";
     15. private bool \_licensed = false;
     16. private string \_price;
     17. public string GroupTitle
     18. {
     19. set
     20. {
     21. if (value != "Italian")
     22. \_licensed = true;
     23. else if (CurrentAppSimulator.LicenseInformation.ProductLicenses[\_name].IsActive)
     24. \_licensed = true;
     25. else
     26. {
     27. CurrentAppSimulator.LicenseInformation.LicenseChanged += OnLicenseChanged;
     28. GetListingInformationAsync();
     29. }
     30. }
     31. }
     32. private async void GetListingInformationAsync()
     33. {
     34. var listing = await CurrentAppSimulator.LoadListingInformationAsync();
     35. \_price = listing.ProductListings[\_name].FormattedPrice;
     36. }
     38. private void OnLicenseChanged()
     39. {
     40. if (CurrentAppSimulator.LicenseInformation.ProductLicenses[\_name].IsActive)
     41. {
     42. \_licensed = true;
     43. CurrentAppSimulator.LicenseInformation.LicenseChanged -= OnLicenseChanged;
     44. if (PropertyChanged != null)
     45. {
     46. PropertyChanged(this, new PropertyChangedEventArgs("IsLicensed"));
     47. PropertyChanged(this, new PropertyChangedEventArgs("IsTrial"));
     48. }
     49. }
     50. }
     52. public bool IsLicensed
     53. {
     54. get { return \_licensed; }
     55. }
     56. public bool IsTrial
     57. {
     58. get { return !\_licensed; }
     59. }
     60. public string FormattedPrice
     61. {
     62. get
     63. {
     64. if (!String.IsNullOrEmpty(\_price))
     65. return "Purchase Italian Recipes for " + \_price;
     66. else
     67. return "Purchase Italian Recipes";
     68. }
     69. }
     70. }
     71. }
     72. **Note:** ProductLicenseDataSource is similar to the AppLicenseDataSource class you added earlier. Whereas AppLicenseDataSource encapsulates data regarding the licensing state of the app, ProductLicenseDataSource encapsulates information regarding the state of the product named “ItalianRecipes.” That product is defined in license.xml.
  3. Open ItemDetailPage.xaml and add the following statements to the <Page.Resources> section:
     1. XAML
     2. <local:ProductLicenseDataSource x:Key="License" />

<common:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>

* 1. Find the two TextBlocks whose Text property is “{Binding Directions}”. (There are two of them because there’s one for each of three different view states.) Replace each TextBlock with the following statements:
     1. XAML
     2. <TextBlock FontSize="20" FontWeight="Light" Text="{Binding Directions}" TextWrapping="Wrap" Visibility="{Binding IsLicensed, Source={StaticResource License}, Converter={StaticResource BooleanToVisibilityConverter }}" />
     3. <Button Width="225" Height="120" Background="#30ffffff" Click="OnPurchaseProduct" Visibility="{Binding IsTrial, Source={StaticResource License}, Converter={StaticResource BooleanToVisibilityConverter }}">
     4. <Button.Content>
     5. <TextBlock Text="{Binding FormattedPrice, Source={StaticResource License}}" TextWrapping="Wrap" TextAlignment="Center" />
     6. </Button.Content>
     7. </Button>
  2. Open ItemDetailPage.xaml.cs and add the following using statement at the top of the file:
     1. C#
     2. using Windows.ApplicationModel.Store;
  3. Add the following statements to the end of the LoadState method:
     1. C#
     2. // Pass the group title to the LicenseDataSource (important!)
     3. ProductLicenseDataSource license = (ProductLicenseDataSource)this.Resources["License"];
     4. license.GroupTitle = item.Group.Title;
     5. **Note:** This code is “important” because it lets ProductLicenseDataSource know whether the recipe currently displayed is an Italian recipe or another recipe.
  4. Add the following method to ItemDetailPage.xaml.cs to request a product purchase when the purchase button is clicked:
     1. C#
     2. private void OnPurchaseProduct(object sender, RoutedEventArgs e)
     3. {
     4. // Purchase the "ItalianRecipes" product
     5. CurrentAppSimulator.RequestProductPurchaseAsync("ItalianRecipes",true);
     6. }

Task 2 – Make a Product Purchase

All that remains now is to test your changes and see a product purchase in action.

* 1. Press F5 to launch the application.
  2. Tap one of the Italian recipes to go to the item-detail page.
  3. Confirm that a purchase button appears in place of the cooking directions, as shown in Figure 3.
  4. 
  5. Figure 3
  6. UI for making a product purchase
  7. Tap the button to initiate a product purchase.
  8. Tap Continue in the Windows Store dialog to simulate a product purchase.
  9. Confirm that the button disappears and cooking directions appear in its place.
  10. View some of the other Italian recipes and verify that cooking directions are shown as normal.
  11. Return to Visual Studio and stop debugging.

Summary

The exercises you performed in this lab demonstrate some of the most important aspects of the Windows Store API – how to detect trial versions, how to simulate purchases of an application, how to simulate in-app product purchases, and how to retrieve information about those products. Of course, in a real application, you’ll want to replace calls to CurrentAppSimulator with calls to CurrentApp. With that, you’ll have all the tools you need to monetize your app. Go forth and generate revenue!