

Hands-on lab

Lab 3: Searching and sharing

September 2012



Contents

[Overview 3](#_Toc334624163)

[Exercise 1: Add recipe sharing 4](#_Toc334624164)

[Task 1 – Invoke the Share charm 4](#_Toc334624165)

[Task 2 – Implement recipe sharing 4](#_Toc334624166)

[Task 3 – Implement recipe image sharing 6](#_Toc334624167)

[Exercise 2: Add recipe search 8](#_Toc334624168)

[Task 1 – Invoke the Search charm 8](#_Toc334624169)

[Task 2 – Add search support 9](#_Toc334624170)

[Task 3 – Add search suggestions 12](#_Toc334624171)

[Task 4 – Add support for external launching 13](#_Toc334624172)

[Summary 15](#_Toc334624173)

Overview

* 1. One of the key features of a Windows 8 user experience is the use of charms that slide in from the right side of the screen in response to swipes or presses of Windows logo key+C. These buttons (“charms”) provide a means for Windows Store apps to expose commonly used features in a manner that is consistent across apps. For example, if you want to perform a search in an app, you select the Search charm and type a search term into the search pane. The UI—and the actions required to invoke that UI—are the same in every app. To share data with another app, you use the Share charm. An app that supports sharing can then share data. For example, a paint app can share a drawing, or Contoso Cookbook can share a recipe, with other apps that support sharing.
  2. In this lab, you will add support for searching and sharing to Contoso Cookbook. You will get first-hand experience implementing Search and Share contracts. You will also learn how these contracts provide a higher level of integration, between two apps or between an app and Windows itself.

# Objectives

* 1. This lab will show you how to:
  + Implement sharing in a Windows Store app.
  + Implement search in a Windows Store app.
  + Implement search suggestions.

# System requirements

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

# Setup

* 1. To prepare your computer for this lab, you must:
  2. Install Windows 8.
  3. Install Microsoft Visual Studio 2012.

# Exercises

* 1. This Hands-on lab includes the following exercises:
  2. Add recipe sharing
  3. Add recipe search
  4. Estimated time to complete this lab: **30 to 40 minutes**.

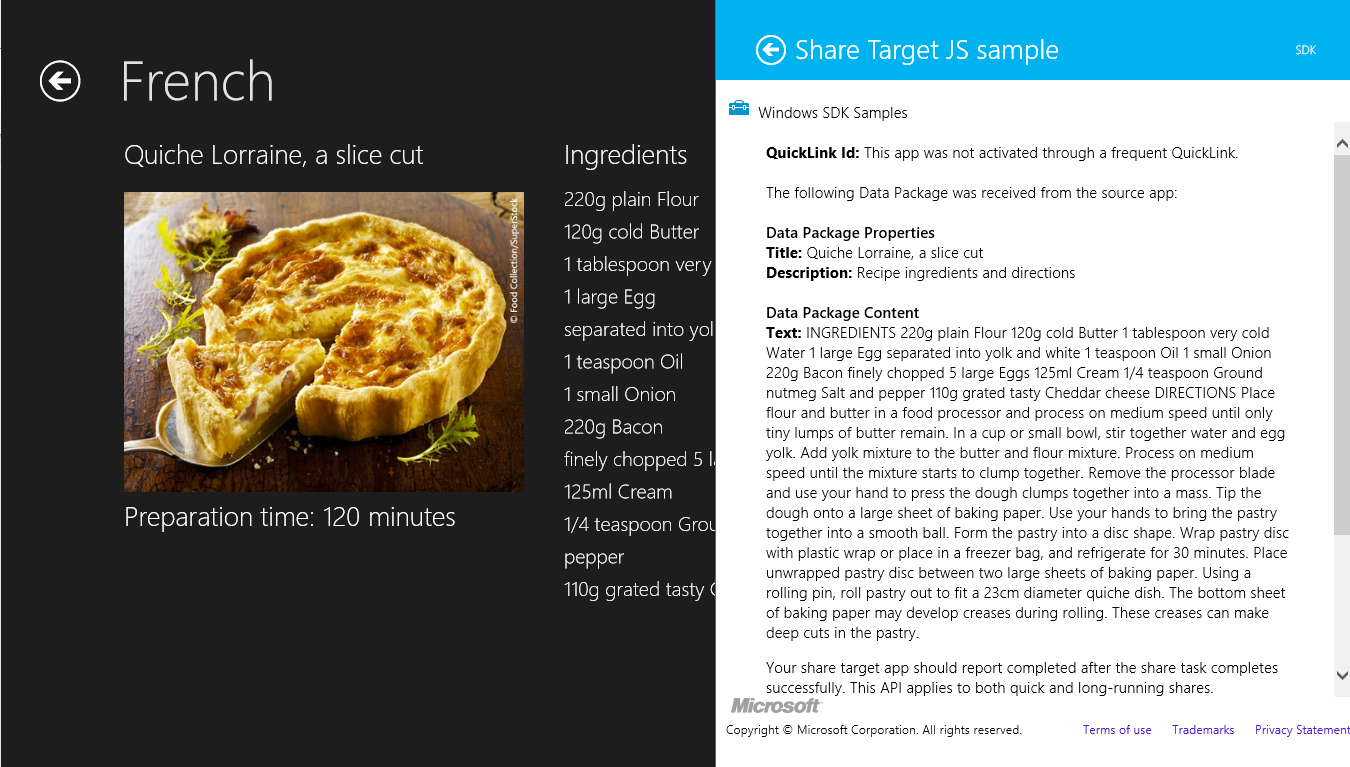
Exercise 1: Add recipe sharing

1. In Exercise 1, you will add sharing support to Contoso Cookbook so that recipes can be shared with other apps. You will share two types of data for each recipe: textual data that includes the recipe name, ingredients, and directions, and image data that contains a pictorial representation of the recipe.

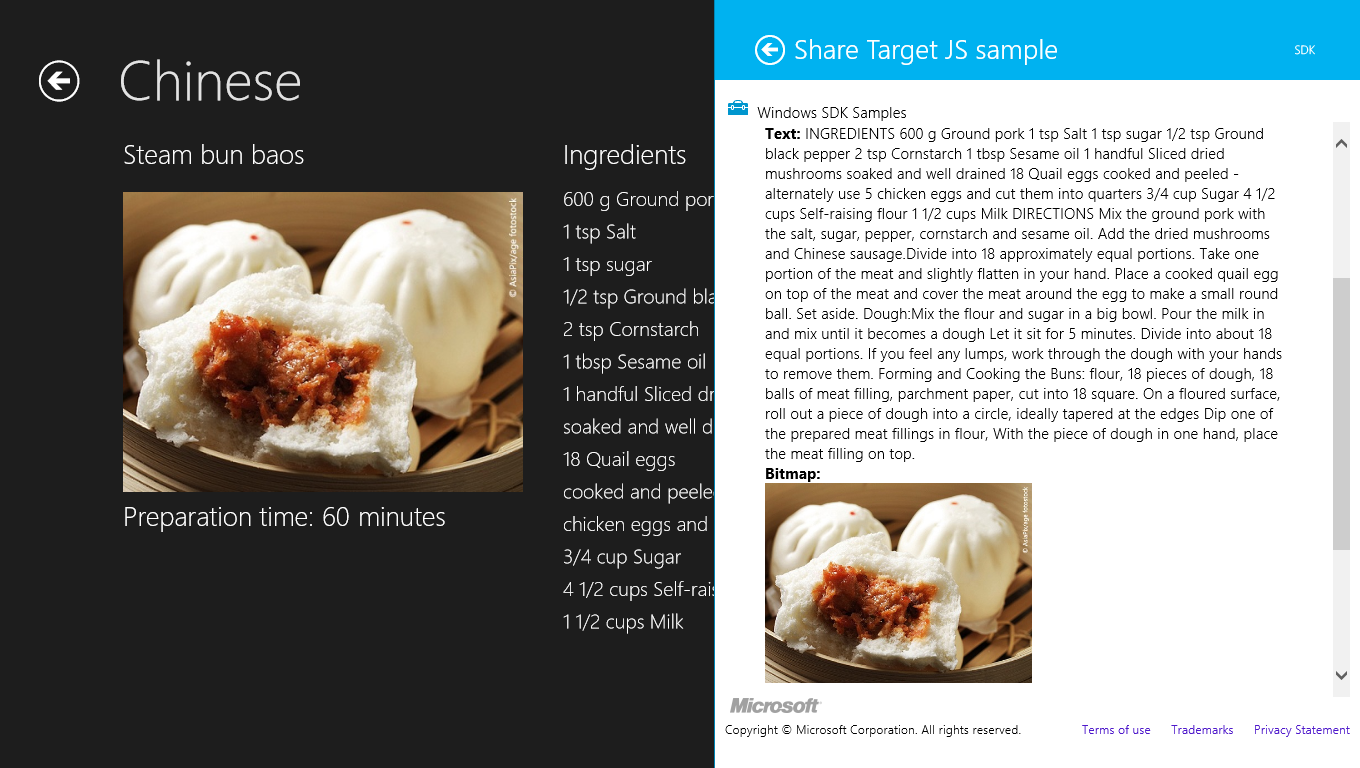
Task 1 – Invoke the Share charm

* 1. To get started, let’s see how the Share charm behaves when it’s invoked in Contoso Cookbook before sharing support is added.
  2. In Visual Studio, open the ContosoCookbook project you finished in Lab 2. If you did not complete Lab 2 or would like to start with a reference copy, you will find a completed version of the lab in the starting materials.
  3. Press F5 to launch Contoso Cookbook.
  4. Tap one of the recipes to show the item-detail page.
  5. Display the charms by swiping from right to left from the right edge of the screen, or by pressing Windows logo key+C.
  6. Tap the Share charm to display the sharing pane.
  7. Since Contoso Cookbook currently does not implement a sharing contract, the sharing pane informs you “This app can’t share.”
  8. Return to Visual Studio and stop debugging.

Task 2 – Implement recipe sharing

* 1. Now that you have seen what the sharing pane looks like when an app does not support sharing, let’s add a Share contract to Contoso Cookbook so it can share recipe data. First, we’ll need to add a bit of infrastructure to support that contract.
  2. Open ItemDetailPage.xaml.cs and add the following using statements at the top of the file.
     1. C#
     2. using Windows.ApplicationModel.DataTransfer;
     3. using System.Text;
     4. using Windows.Storage.Streams;
  3. Find the LoadState method and add the following statements to the end of it.
     1. C#
     2. // Register for DataRequested events
     3. DataTransferManager.GetForCurrentView().DataRequested += OnDataRequested;
  4. Now add the following method to the ItemDetailPage class.
     1. C#
     2. void OnDataRequested(DataTransferManager sender, DataRequestedEventArgs args)
     3. {
     4. var request = args.Request;
     5. var item = (RecipeDataItem)this.flipView.SelectedItem;
     6. request.Data.Properties.Title = item.Title;
     7. request.Data.Properties.Description = "Recipe ingredients and directions";
     8. // Share recipe text
     9. var recipe = "\r\nINGREDIENTS\r\n";
     10. recipe += String.Join("\r\n", item.Ingredients);
     11. recipe += ("\r\n\r\nDIRECTIONS\r\n" + item.Directions);
     12. request.Data.SetText(recipe);
     13. }
     14. **Note:** You implement sharing when you register a handler for **DataTransferManager**’s **DataRequested** events. These events fire when the user activates the Share charm. In this example, you respond to that event by calling **SetText** on the **DataPackage** object, which is exposed through **args.Request.Data**, to provide the recipe in the form of text. When the sharing pane appears with a list of share targets, the list includes only share targets that can consume text.
  5. Find the SaveState method and add the following statements to the end of it.
     1. XAML
     2. // Deregister the DataRequested event handler
     3. DataTransferManager.GetForCurrentView().DataRequested -= OnDataRequested;
  6. Press F5 to launch the app.
  7. Tap one of the recipes to show the recipe-detail page.
  8. Display the charms and select the Share charm to display the sharing pane. The sharing pane now shows you a list of *share targets* installed on your device – apps that can consume data shared by share sources.
     1. **Note:** If you have not done so already, now would be a great time to install the Share Target Sample App that comes with the Windows 8 SDK Samples pack. The Share Target Sample App demonstrates how to write share-target apps. More importantly, it provides a sharing target to test with, as you develop apps that act as sharing sources. It accepts images as well as text and other data types. To install the sample, open it in Visual Studio and run it one time. After that, it should appear in the list of share targets when you select the Share charm from any Windows Store app.
  9. Select one of the share targets listed in the sharing pane and verify that it received the recipe data. Figure 1 shows the Share Target Sample App after it accepts recipe text shared by Contoso Cookbook.
     1. 
     2. Figure
     3. Share Target Sample App showing a shared recipe
  10. Return to Visual Studio and stop debugging.

Task 3 – Implement recipe image sharing

* 1. Contoso Cookbook can now share textual recipe data, but since each recipe is accompanied by an image, it should share recipe images as well. That way, a share target that accepts images can show a photo of the recipe along with the recipe text (assuming the share target supports text as well as images). Let’s revise your sharing code to support bitmap images as well as text.
  2. Return to the OnDataRequested method you added in the previous task.
  3. Add the following statements to the end of the method.
     1. C#
     2. // Share recipe image
     3. var reference = RandomAccessStreamReference.CreateFromUri(new Uri(item.ImagePath.AbsoluteUri));
     4. request.Data.Properties.Thumbnail = reference;
     5. request.Data.SetBitmap(reference);
  4. Press F5 to launch the app.
  5. Tap one of the recipes to show the recipe-detail page.
  6. Display the charms and select the Share charm to display the sharing pane.
  7. Select one of the share targets listed in the sharing pane and verify that it received the recipe image. Figure 2 shows the Share Target Sample App after it accepts a recipe image from Contoso Cookbook.
     1. 
     2. Figure
     3. Share Target Sample App showing a shared recipe image
  8. Return to Visual Studio and stop debugging.

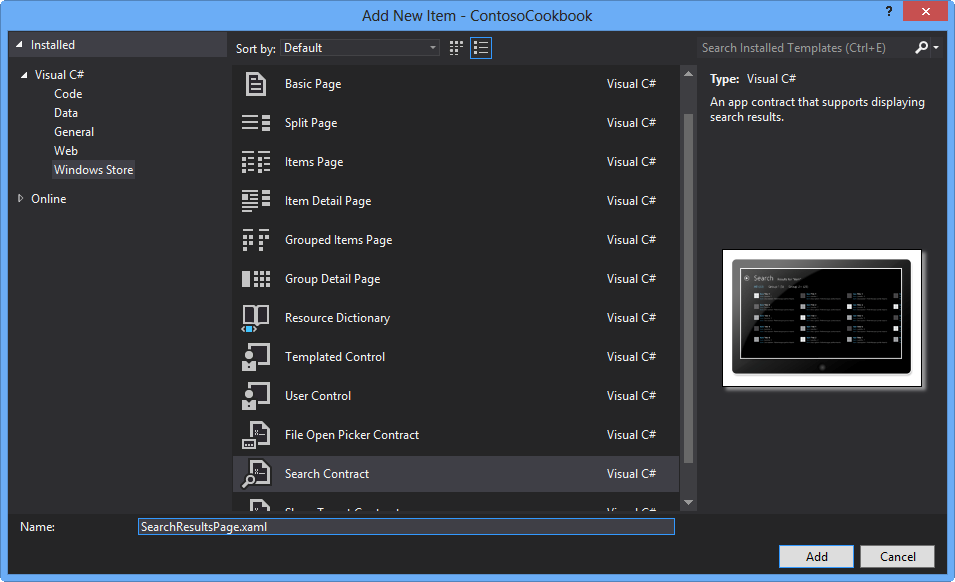
Exercise 2: Add recipe search

1. In Exercise 2, you will add search support to Contoso Cookbook so users can use the Search charm to search for recipe data. For example, a user who wants to find all recipes that contain sugar will invoke the Search charm, type “sugar” into the search box, and be presented with a list of sugary recipes.

Task 1 – Invoke the Search charm

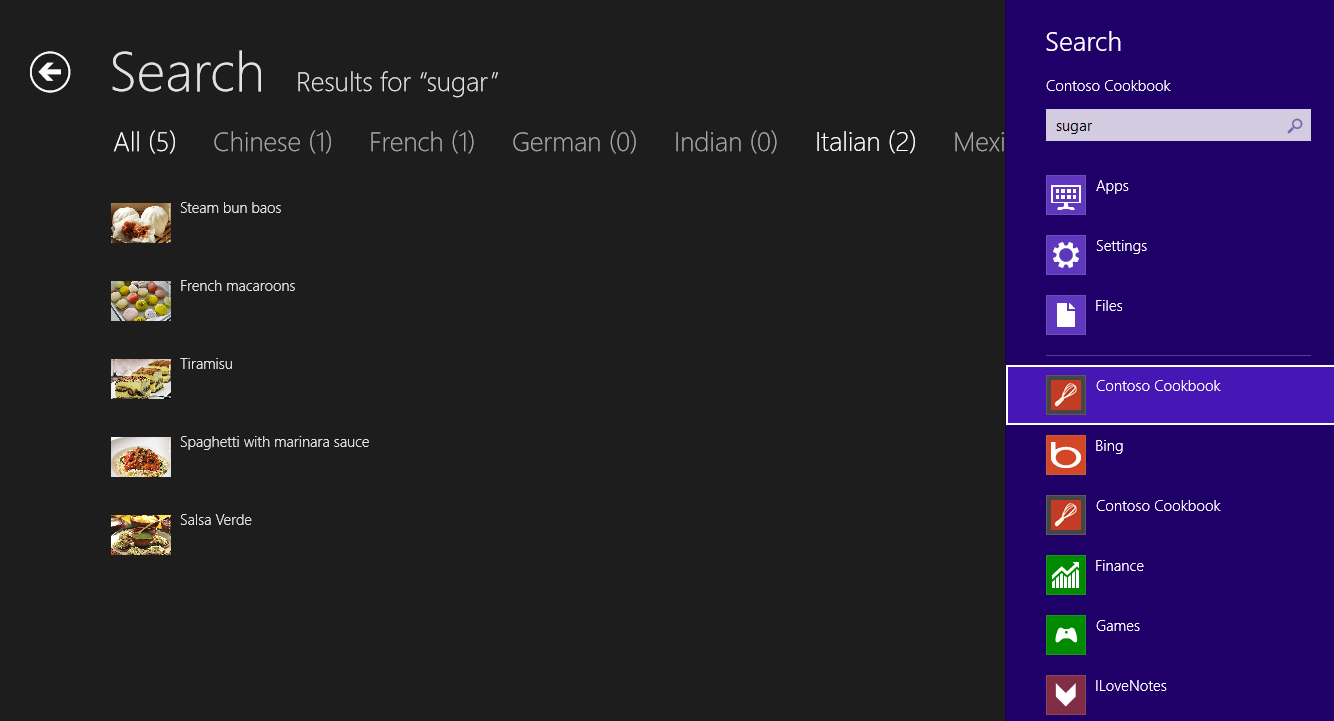
* 1. Before adding search support to Contoso Cookbook, let’s see what the search UI looks like when it’s invoked while Contoso Cookbook is the foreground app.
  2. Press F5 to launch the app.
  3. Display the charms by swiping from right to left from the right edge of the screen, or by pressing Windows logo key+C.
  4. Tap the Search charm to display the search pane.
  5. Type “sugar” (without the quotation marks) into the search box and press Enter, or tap the magnifying-glass icon at the right end of the search box.
  6. Windows tells you “This app can’t be searched.” That will change once you add search support to Contoso Cookbook.
  7. Return to Visual Studio and stop debugging.

Task 2 – Add search support

* 1. Now let’s implement the Search contract in Contoso Cookbook. Visual Studio will do most of the work for you by inserting a C# contract into your app. You will need to tweak the code to perform domain-specific searches within your app’s data. It’s easy to do, as the next few steps will demonstrate.
  2. Right-click the project in Solution Explorer and use the **Add > New Item** command to add a Search contract named SearchResultsPage.xaml, as shown in Figure 3.
     1. 
     2. Figure 3
     3. Adding a Search contract
  3. In the <Pages.Resources> section of SearchResultsPage.xaml, change “My Application” to “Search” in the string resource named “AppName” to change the title at the top of the page.
     1. XAML
     2. <x:String x:Key="AppName">Search</x:String>
  4. Add an ItemClick=”OnItemClick” attribute to the GridView named “resultsGridView” and to the ListView named “resultsListView.” The GridView shows search results when the app isn’t snapped, and the ListView shows search results when the app *is* snapped.
  5. Open SearchResultsPage.xaml.cs and add the following method to the SearchResultsPage class, to navigate to the recipe page when an item is clicked in the search results page.
     1. C#
     2. private void OnItemClick(object sender, ItemClickEventArgs e)
     3. {
     4. // Navigate to the page showing the recipe that was clicked
     5. this.Frame.Navigate(typeof(ItemDetailPage), ((RecipeDataItem)e.ClickedItem).UniqueId);
     6. }
  6. Add the following using statement at the top of SearchResultsPage.xaml.cs.
     1. C#
     2. using ContosoCookbook.Data;
  7. Next, add the following field to the SearchResultsPage class.
     1. C#
     2. // Collection of RecipeDataItem collections representing search results
     3. private Dictionary<string, List<RecipeDataItem>> \_results = new Dictionary<string, List<RecipeDataItem>>();
  8. Go to the LoadState method and add the following statements *before* the comment that reads “Communicate results through the view model.”
     1. C#
     2. // Search recipes and tabulate results
     3. var groups = RecipeDataSource.GetGroups("AllGroups");
     4. string query = queryText.ToLower();
     5. var all = new List<RecipeDataItem>();
     6. \_results.Add("All", all);
     7. foreach (var group in groups)
     8. {
     9. var items = new List<RecipeDataItem>();
     10. \_results.Add(group.Title, items);
     12. foreach (var item in group.Items)
     13. {
     14. if (item.Title.ToLower().Contains(query) || item.Directions.ToLower().Contains(query))
     15. {
     16. all.Add(item);
     17. items.Add(item);
     18. }
     19. }
     20. filterList.Add(new Filter(group.Title, items.Count, false));
     21. }
     22. filterList[0].Count = all.Count;

**Note:** The code you just added searches all recipe titles and directions for the query text entered by the user. For each matching recipe it finds, it adds a RecipeDataItem to a list of all recipes, and a RecipeDataItem to a list of recipes in a specific group. The lists are maintained in the dictionary you declared a moment ago and are tagged with group names such as “All,” “Chinese,” and so on.

This code also populates the filter list (represented by the filterList variable) at the top of the search results page, with group names that show the number of matching recipes in each group.

* 1. Find the Filter\_SelectionChanged method and add the following statement to the if clause (right after the // TODO: Respond to the change in active filter comment).
     1. C#
     2. this.DefaultViewModel["Results"] = \_results[selectedFilter.Name];
  2. Open StandardStyles.xaml in the Common folder.
  3. Find the **DataTemplate** resource named “StandardSmallIcon300x70ItemTemplate.” Remove the TextBlock bound to the **Subtitle** property. Also change the width and height properties of the Border, and bind the first TextBlock ‘s **Text** property to ShortTitle rather than Title.
     1. XAML
     2. <DataTemplate x:Key="StandardSmallIcon300x70ItemTemplate">
     3. <Grid Width="294" Margin="6">
     4. <Grid.ColumnDefinitions>
     5. <ColumnDefinition Width="Auto"/>
     6. <ColumnDefinition Width="\*"/>
     7. </Grid.ColumnDefinitions>
     8. <Border Background="{StaticResource ListViewItemPlaceholderBackgroundThemeBrush}" Margin="0,0,0,10" Width="60" Height="45">
     9. <Image Source="{Binding Image}" Stretch="UniformToFill"/>
     10. </Border>
     11. <StackPanel Grid.Column="1" Margin="10,-10,0,0">
     12. <TextBlock Text="{Binding ShortTitle}" Style="{StaticResource BodyTextStyle}" TextWrapping="NoWrap"/>
     13. <TextBlock Text="{Binding Description}" Style="{StaticResource BodyTextStyle}" Foreground="{StaticResource ApplicationSecondaryForegroundThemeBrush}" TextWrapping="NoWrap"/>
     14. </StackPanel>
     15. </Grid>
     16. </DataTemplate>
  4. Press F5 to start the app.
  5. Display the charms.
  6. Tap the Search charm to display the search pane.
  7. Type “sugar” (without the quotation marks) into the search box at the top of the search pane and press Enter, or tap the magnifying-glass icon at the right end of the search box.
  8. Verify that five recipes appear in the search results (see Figure 4).
     1. 
     2. Figure 4
     3. Search results for “sugar”
  9. Select one of the recipes and verify that the recipe detail appears.
  10. Return to Visual Studio and stop debugging.

Task 3 – Add search suggestions

* 1. A useful enhancement that you can add to the search experience is to provide suggestions as the user types a search term into the search box. It’s easy to do; all you have to do is register a handler for SuggestionsRequested events. Here’s how.
  2. Open App.xaml.cs and add the following using statement at the top of the file.
     1. C#
     2. using Windows.ApplicationModel.Search;
  3. Add the following statements to the beginning of the **OnLaunched** method, right after the call to RecipeDataSource.LoadLocalDataAsync or RecipeDataSource.LoadRemoteDataAsync to load recipe data.
     1. C#
     2. // Register handler for SuggestionsRequested events from the search pane
     3. SearchPane.GetForCurrentView().SuggestionsRequested += OnSuggestionsRequested;
  4. Now add the following method to the **Application** class.
     1. C#
     2. void OnSuggestionsRequested(SearchPane sender, SearchPaneSuggestionsRequestedEventArgs args)
     3. {
     4. string query = args.QueryText.ToLower();
     5. string[] terms = { "salt", "pepper", "water", "egg", "vinegar", "flour", "rice", "sugar", "oil" };
     7. foreach(var term in terms)
     8. {
     9. if (term.StartsWith(query))
     10. args.Request.SearchSuggestionCollection.AppendQuerySuggestion(term);
     11. }
     12. }
     13. **Note:** The code you just added provides search suggestions for words that pattern-match salt, pepper, water, egg, vinegar, flour, rice, sugar, and oil. If the user types “sa,” the word “salt” will appear in the search pane as a suggested completion term. Of course, you can add as many suggestions as you’d like. If you want “ketchup” to appear when the user types “ke,” simply add that term to the list.
  5. Start the app again and type “su” into the search box. Verify that “sugar” appears in the suggestion list underneath the search box, as shown in Figure 5.
     1. 
     2. Figure 5
     3. Search suggestions in action
  6. Return to Visual Studio and stop debugging.

Task 4 – Add support for external launching

* 1. You are almost done, but there is one task left to perform. Currently, search works great if Contoso Cookbook is running when you execute a search. But it does not work at all if the user invokes Search from outside the app. To see for yourself, make sure Contoso Cookbook is not running (you can check for it in Task Manager and terminate it if needed). Then invoke the Search charm, type “sugar” into the search box, and tap Contoso Cookbook in the list of apps in the search pane. You are notified that no results match your search. Let’s fix that.
  2. Open App.xaml.cs and find the **OnSearchActivated** method.
     1. **Note: OnSearchActivated** is a key virtual method in the **Application** class. It’s called to activate the search-results page when the user performs a search, and was added to App.xaml.cs when you added a Search contract to the project. **OnSearchActivated** is called in lieu of **OnLaunched** if the operating system launches Contoso Cookbook from the search pane. If this occurs, we need to reinitialize the app before we activate the search-results page.
  3. Add the following statements to the beginning of the **OnSearchActivated** method.
     1. C#
     2. // Reinitialize the app if a new instance was launched for search
     3. if (args.PreviousExecutionState == ApplicationExecutionState.NotRunning ||
     4. args.PreviousExecutionState == ApplicationExecutionState.ClosedByUser ||
     5. args.PreviousExecutionState == ApplicationExecutionState.Terminated)
     6. {
     7. // Load recipe data
     8. await RecipeDataSource.LoadLocalDataAsync();
     9. // Register handler for SuggestionsRequested events from the search pane
     10. SearchPane.GetForCurrentView().SuggestionsRequested += OnSuggestionsRequested;
     12. }
     13. **Note:** If you wish to load recipe data from the cloud rather than from local resources, replace the call to LoadLocalDataAsync in the code above with a call to LoadRemoteDataAsync – just as you did in Lab 1.
  4. Use Task Manager to make sure Contoso Cookbook is not running. End the process if it is.
  5. Go to the start screen. Display the charms and tap the Search charm.
  6. Enter a search term such as “sugar.” Then tap Contoso Cookbook in the list of apps in the search pane.
  7. Verify that Contoso Cookbook starts up and displays the correct search results.
  8. Press Alt-F4 to close the app.

Summary

* 1. Contracts are an important part of Windows 8 because they enable apps to integrate with the operating system and they provide a user experience that is both consistent and predictable across apps. This loosely coupled collaboration is very synergistic and extensible, allowing you to share anything with any app, search within any app, and so on.
  2. In this lab, you learned about two types of contracts first-hand: Share contracts and Search contracts. In a later lab, you will use another type of contract to integrate Settings. But first there’s something else we need to tackle: media capture, using the cameras built into most PCs and mobile devices today. That’s the subject of the next lab, so let’s keep moving!