# Searching for Search

In this blog we will discuss the new recommendation for doing search within Windows 8.1. For reference this blog is based off the build session “3-144 Design and Build a Great Search Experience in Your App” found at this location: <http://channel9.msdn.com/Events/Build/2013/3-144>.

Before we dive into any of the code I’ll pass along a few of the “rules” that I gathered from the session:

* Users love search when it’s efficient
* One search method should appear in the application – dedicated to the application
* Search should be optimized to get you to where you want to go – FAST.

On a side note…

One important point that was mentioned during the session – that you will want to remember in Windows 8.1: **One search method should appear in the application – dedicated to the application**.

This is a bit of a change from Windows 8 when we were told to use the Search charm within our own applications for searching. In Windows 8.1 swiping the side bar and then clicking on the search charm will result in search activating, however with a context of “Everywhere”. One can change the context to the app that is currently being used – however the default is everywhere.

In the case of Windows 8.1, the directive is to provide a Search box in the actual application or have a search button where the application can activate the search charm programmatically and set the default context to the local application. Windows 8.1 has provided us a SearchBox control where in Windows 8 we would use the Search Pane and the charm.

The following example of the new Win8.1 travel application shows the new application search entry field.

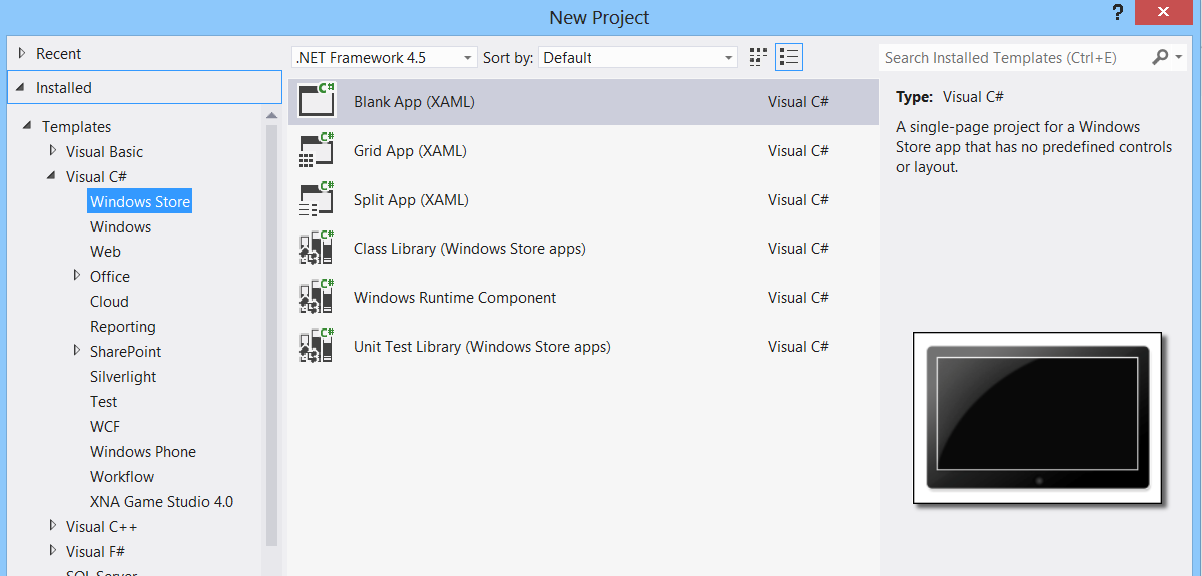


The point of this blog was to integrate search into our Win 8.1 application – so let’s get coding. In this code sample we will integrate a search box that has all the efficiency features from Windows 8 search pane:

* History suggestions
* Query and results suggestions
* Type-to-search

# Project Setup

We will create a very simple C# Windows Store application and add some search to it. From within Visual Studio 2013, create a new C# Windows Store Application



In the XAML add the SearchBox Control:

<SearchBox x:Name="SearchBoxSuggestions"

SearchHistoryEnabled="False"

Width="300"

SuggestionsRequested="SearchBoxEventsSuggestionsRequested"

QuerySubmitted="SearchBoxEventsQuerySubmitted"/>

In the code behind add the following event handlers for the click events for the buttons:

private void SearchBoxEventsSuggestionsRequested(

SearchBox sender,

SearchBoxSuggestionsRequestedEventArgs e)

Note: I know – where’s the MVVM, the commands, etc. This exercise is about using the new SearchBox control.

We’ll demonstrate the Query Suggestions to the Search Control. Let’s add some items that we can search through first. In this example we’ll search for a fruit.

Create a property and add some fruit:

private List<String> fruits;

fruits = new List<string>();

fruits.Add("Apple");

fruits.Add("Banana");

fruits.Add("Cherry");

fruits.Add("Clementine");

fruits.Add("Dragon Fruit");

fruits.Add("Orange");

fruits.Add("Plum");

In the SearchBoxEventsSuggestionsRequested event we will need to add some code to populate the query suggestions. The query suggestions will be taken from the list of fruits. This event is fired as the user types and the application has an opportunity to use the text being entered to create Query Suggestions and even Result Suggestions.

For the Query suggestions we will go through the list of fruits to and pull out the ones that match the current query text:

private void SearchBoxEventsSuggestionsRequested(SearchBox sender,

SearchBoxSuggestionsRequestedEventArgs e)

{

string queryText = e.QueryText;

Windows.ApplicationModel.Search.SearchSuggestionCollection suggestionCollection =

e.Request.SearchSuggestionCollection;

foreach (string suggestion in fruits)

{

if (suggestion.StartsWith(queryText, StringComparison.CurrentCultureIgnoreCase))

{

suggestionCollection.AppendQuerySuggestion(suggestion);

}

}

}

The event SearchBoxEventsQuerySubmitted will be fired when a user submits a search query. In this case we will add it to a search results list, just to see some results.

private void SearchBoxEventsQuerySubmitted(SearchBox sender,

SearchBoxQuerySubmittedEventArgs e)

{

\_searchResults.Add(e.QueryText);

SearchBoxSuggestions.QueryText = "";

}

If you start typing you’ll probably notice that no history is displayed. It is recommended that you use history, but also provide the user with a way to clear the history.

To enable the history change the **SearchHistoryEnabled** to True:

<SearchBox x:Name="SearchBoxSuggestions"

SearchHistoryEnabled="True"

Width="300"

SuggestionsRequested="SearchBoxEventsSuggestionsRequested"

QuerySubmitted="SearchBoxEventsQuerySubmitted"/>

And you can use the following code to clear the history:

Windows.ApplicationModel.Search.Core.SearchSuggestionManager searchman =

new Windows.ApplicationModel.Search.Core.SearchSuggestionManager();

searchman.ClearHistory();