Search, Share, and Tiles

In this lab, we will code in bursts. Get the starting source code and follow along. I will commit changes after every burst, so you can get caught up if you get lost.

## Gitting the Source Code

If you have a git client installed, then clone the repository. Create a working branch. For example:

cd c:\projects

git clone git://github.com/michaellperry/win8unleashed\_myblog.git

cd Composition

git checkout –b take1

After each successful burst, commit your changes. If you ever get lost, commit that branch, go back to master, and create a new one:

git add –A

git commit –m "I missed that."

git checkout master

git pull

git checkout –b take2

## Downloading the Source Code

If you don’t have git installed, then go to the following URL and click on the “ZIP” button:

https://github.com/michaellperry/win8unleashed\_myblog

Unzip to your project folder. If you ever get lost, go back to the web page and download the zip again.

# Tiles

Windows 8 live tiles are a great way to expose information to your users. This will entice them to come back to your application. You can update the application’s tile either by push notification, or from the application itself. In this lab, we’ll perform the update in the app.

Tile updates are sent to the OS as XML files. But you don’t need to know the schema. You can use the helper classes that are included in the project source code.

We’ll use the TileContentFactory to create … tile content! We have several formats to choose from. Let’s create a wide tile. Open up the TileManager and add this code:

var tileContent = TileContentFactory.CreateTileWideImageAndText01();

tileContent.TextCaptionWrap.Text = article.Title;

tileContent.Image.Src = article.ImagePath;

tileContent.Image.Alt = article.Subtitle;

If the user has chosen a small tile, we’ll show different content:

var squareContent = TileContentFactory.CreateTileSquareImage();

squareContent.Image.Src = article.ImagePath;

squareContent.Image.Alt = article.Title;

tileContent.SquareContent = squareContent;

Last step, turn the content into XML and tell the OS to update the tile:

TileNotification notification = tileContent.CreateNotification();

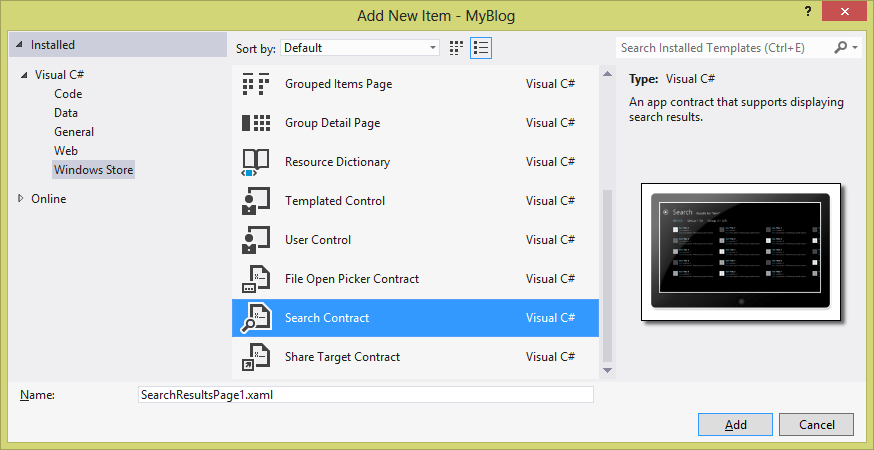
TileUpdateManager

.CreateTileUpdaterForApplication()

.Update(notification);

# Search

Application search is available from the charms, even when the app is not running. The app needs to declare its search capability. The easiest way to do this is to right-click the project, add item, select Windows Store, and add a Search Contract.



This creates a view for the search results. The App.OnSearchActivated() method is called when the user performs the search. It displays the search results page. The LoadState method is overridden, and receives the query text as the navigation parameter. Let’s capture this string.

private string \_queryText;

protected override void LoadState(Object navigationParameter, Dictionary<String, Object> pageState)

{

\_queryText = navigationParameter as String;

Add code to Filter\_SelectionChanged to run the query.

string lowerQueryText = \_queryText.ToLower();

var matchingArticles =

from g in SampleDataSource.GetGroups("AllGroups")

from a in g.Items

where a.Title.ToLower().Contains(lowerQueryText)

|| a.Description.ToLower().Contains(lowerQueryText)

|| a.Content.ToLower().Contains(lowerQueryText)

select a;

this.DefaultViewModel["Results"] = matchingArticles.ToList();

Now the user can search for articles whether they are in the app or not.

# Share

For a user to get the most out of our app, they should be able to share data with other apps. It can be a share target, a share source, or both. In this lab, we’ll make our app a share source.

The DataTransferManager coordinates sharing between apps. When the user navigates to the ItemDetailsPage, it will register with the manager.

protected override void OnNavigatedTo(NavigationEventArgs e)

{

var manager = DataTransferManager.GetForCurrentView();

manager.DataRequested += manager\_DataRequested;

base.OnNavigatedTo(e);

}

When they navigate away, it will clean up.

protected override void OnNavigatedFrom(NavigationEventArgs e)

{

var manager = DataTransferManager.GetForCurrentView();

manager.DataRequested -= manager\_DataRequested;

base.OnNavigatedFrom(e);

}

Now when the manager asks for data to share, the view will provide some.

void manager\_DataRequested(DataTransferManager sender, DataRequestedEventArgs args)

{

DataRequest request = args.Request;

var item = flipView.SelectedItem as SampleDataItem;

request.Data.Properties.Title = item.Title;

request.Data.Properties.Description = item.Description;

request.Data.SetText(item.Content);

}

Now you can open up the share charm while looking at an article, and send it to email or any other app that is a share target.