Selfie App Lab

# Introduction

The purpose of this lab is to build a Windows Store App using what we’ve learned so far to create a picture taking app.

The shell code can be found here: <https://github.com/randysingh/bootcamp2015>

Steps to complete the lab.

1. Starting with the MainView we’ll create a welcome page. This will consist of a title, a webview and button to navigate us to the next view. We’ll have our webview navigate to the selfie song YouTube video “<https://www.youtube.com/watch?v=kdemFfbS5H0>”. Navigation will need to be done in the code behind. You’ll also want to handle OnNavigatedTo and OnNavigatedFrom because the webview will continue to play even though you’ve navigated away. You can simply navigate to blank string content to stop it from playing (Yea I know the webview isn’t great ☺)
2. Next you’ll add in a PictureViewModel and PictureView in the correct folders. PageViewModel will need to inherit from Screen and you’ll need to register the ViewModel in App.xaml.cs.
3. Create a delegate command on the MainView, bind it to the “Enter” Button to navigate to the PictureViewModel. (You’ll need to use INavigationService).
4. Create the layout for the PictureView. You’ll probably want to use a grid layout.
5. Add in the required properties for the PictureViewModel:

DelegateCommand GoBackCommand

DelegateCommand TakePictureCommand

DelegateCommand SavePhoto

ObservableCollection<string> Photos

string VisualState

TakeSelfieEvent EventHandler.

1. PictureViewModel will need to have INavigationService and IStorageProvider injected. Use IStorageProvider to populate the Photos when the viewmodel is initialized (override OnInitialize()).
2. Firing the TakePictureCommand should fire the TakeSelfieEvent event if it’s not null.
3. Taking the photo is Win 8 specific code so we’ll need to fire this in the view.

var camera = new CameraCaptureUI();

camera.PhotoSettings.AllowCropping = true;

camera.PhotoSettings.CroppedAspectRatio = new Size(1,1);

var file = await camera.CaptureFileAsync(CameraCaptureUIMode.Photo);

1. In the view we’ll need to add handle the TakeEventHandler event which will fire the camera. Eg.

private void OnDataContextChanged(FrameworkElement sender, DataContextChangedEventArgs args)

{

if (ViewModel != null && ViewModel.TakeSelfieEvent == null)

{

ViewModel.TakeSelfieEvent += TakePicture;

}

}

1. Once the photo is taken we’ll pass it to the view model by firing the SavePhoto delegate command which will save it using the storage provider and we’ll need to repopulate photos.
2. Now that this is mostly complete let’s bind this to the gridview and see it in action. (GridView will need to use the correct ItemsPanel and ItemsTemplate to display this)
3. Now let’s add in some fancy animation when the Picture View is loaded using the VisualState and VisualStateBehavior. Let’s try adding a nice fade in effect. (Or whatever floats your boat, just want to see you use it)
4. Next the gridview allows you to select items, we can also allow you to select multiple. You can bind the selected items to a property on the view model. Then we can add some delete functionality! Add a delete button that only becomes active when there are items selected in the gridview and then wire it up to a DeleteCommand to delete photos. Getting the selected items will need to be done in the code behind.
5. Delete should open a flyout confirmation dialog where the user can click yes or no to delete.
6. Now we have a pretty sweet photo managing app!
7. If you have time you can also connect the photos from your photo library here and view all your photos in one place.

Sample screenshots:



