**FNDev1 Coursework**

THLibrary: Library Search Facility

**Development Notes**

**Table of Contents**

[Project Setup 3](#_Toc342301599)

[Database setup 3](#_Toc342301600)

[UI Layer 3](#_Toc342301601)

[ViewModel 3](#_Toc342301602)

[Windows 8 Store App 4](#_Toc342301603)

[Overview 4](#_Toc342301604)

[Sample screen layout 4](#_Toc342301605)

[Solution Structure 4](#_Toc342301606)

[Development So Far 5](#_Toc342301607)

[Currently Working On 5](#_Toc342301608)

[Components 5](#_Toc342301609)

[Unit Testing 5](#_Toc342301610)

[Development Problems 6](#_Toc342301611)

[Future Development 7](#_Toc342301612)

[Templates for Survey Questions and Responses 7](#_Toc342301613)

# Project Setup

## Database setup

The ‘database’ is a text file containing the CSV records of the books. These are loaded at the application start-up to the Business model and all access within the application; searches etc are performed against this in memory Business Model.

The Searches are also loaded at application start-up: these are held as an XML serialised data file containing the complete list of searches. They are also loaded into an in-memory business model and all access is performed against this model.

## UI Layer

The UI layer is a Windows Store App, The application follows the MVVM pattern with the LibrarySearchViewModel performing the role of the VM. The view model contains ObservableCollections<> of each type needed to support the UI screen and this makes use of the binding facilities offered by the xaml, which the UI screens are written in.

The Model in the MVVM pattern is the Core business model and is accessed via the view model..

## ViewModel

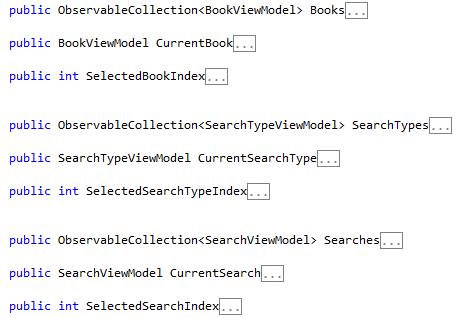


Figure 1 - The View Model so far.

# Windows 8 Store App

## Overview

Application is a single page application, not conforming to exactly the style of program required for a Windows Store app.

## Sample screen layout

This is as far as it’s gone.

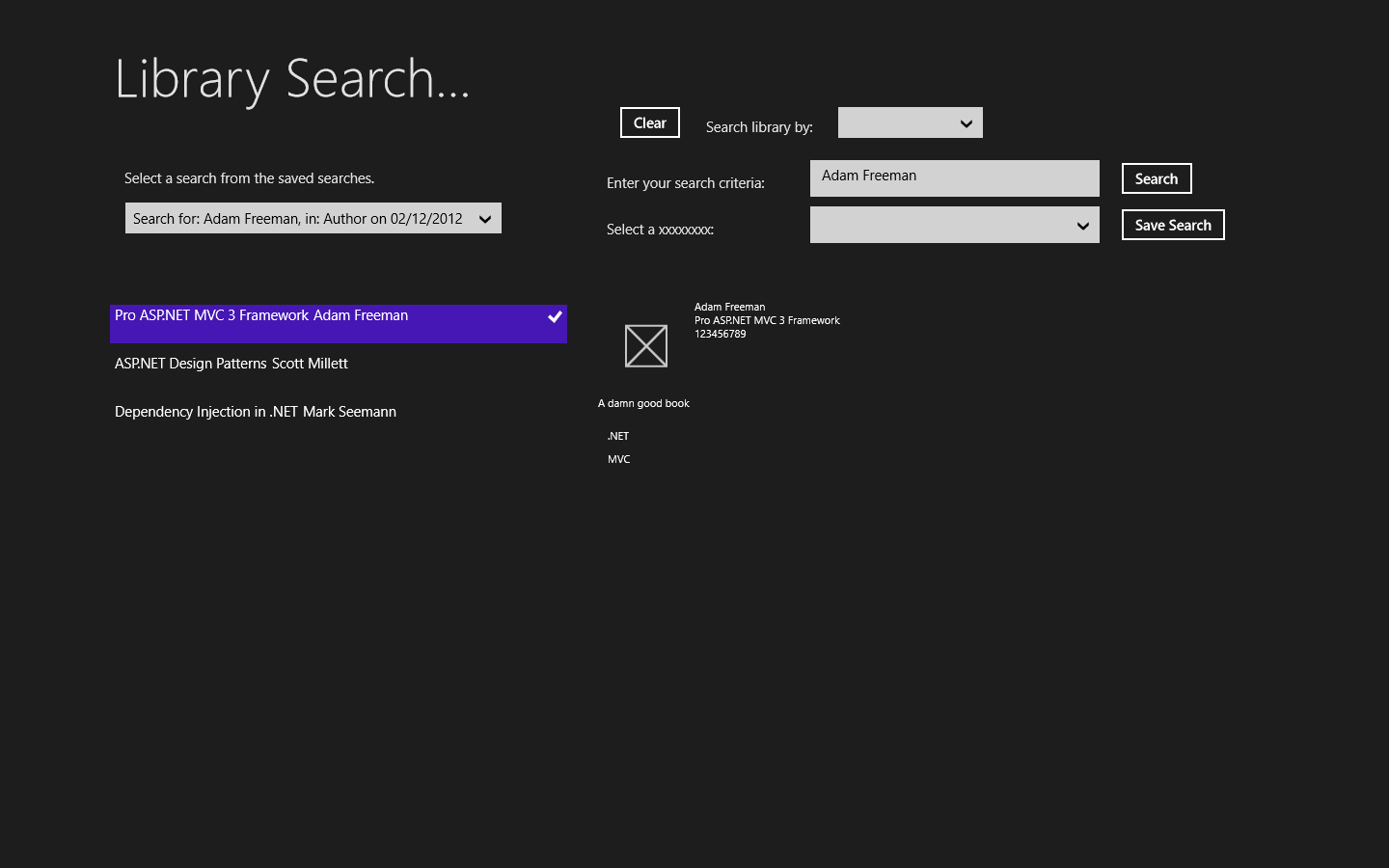


Figure - Sample application page.

The sections are not distinct enough just yet. Perhaps look at putting the section of a lighter background to separate it from the results on the left and the details of each book on the right, both on the lower half of the panel.

## Solution Structure

The solution is structured using the ‘onion model, consisting of a core project which contains the data model and all aspects of the business model required by the UI layer, which is the xaml Windows Store application.

The data access for the application is the responsibility of the Infrastructure project. Communication between layers is only permitted from an outer layer to the inner Core layer. Therefore no direct communication is permitted between the Infrastructure and UI projects.

The exception is the IoC container project which is a DI implementation using Unity and maps the various modules required by the application and their corresponding implementation’s.

## Development So Far

The layout is not finalised by any means.

The Core and Infrastructure projects are coded and the file access stuff is too. The reading of files is tested, but the writing of the Searches to the XML serialised file is not tested yet (there haven’t been any searches to save yet).

The UI layer is testing with mock data in the UI LibrarySearchViewModel. When the functionality of the UI is complete, with its searches, sort’s etc. access to the Core project and the business model will be plugged in.

## Currently Working On

The LibrarySearchViewModel to link up the select of the search to show the CurrentSearch on the search parameters. The project is to connect the Select Search Type from the top to the CurrentSearch also.

Then the click events of the buttons must be activated. This should propagate automatically through the panel contents.

Have the search happen automatically when the contents of the search string are changed, with the values from the drop downs or the value entered directly

Also, have the search for Authors accept suggested values AS AN IDEA ONLY, IT MOGHT NOT WORK well in terms of design.

## Components

MS Unity is used as the IoC container. It is about the only such framework the installs on the Windows 8 Store app platform. It is the latest, pre-release, version as this is the only one available for this platform.

## Unit Testing

No unit tests are set up at the moment, however, there will be as part of this solution .

It will use the version of MSTest that is supplied with VS2012 speciically for the Windows Store App platform. There is not much point looking for any other testing frameworks as the compatibility, as with the ioC Framework, will be an issue at the moment. This is a penalty of use such state of the art products

## Development Problems

### IoC Container

Using Unity as the IoC container is simple enough: configuration is similar to Ninject, but Ninject does not have a release compatible with Windows Store Apps just yet.

However, it is not possible to use constructer injection with xaml. The LibrarySearchViewModel requires a reference to the repositories, but xaml MUST have a parameterless constructer.

Therefore, the repository is injected in the constructer by using the Unity.Container.Resolve<>() method. However, this does not work when referenced directly within the VM constructor. Instead we have to create a Readonly property on the App.xaml file and expose the repositories from these properties. This is then referenced directly from within the ViewModel and this works OK. It is , however, considered a bit of a hack but the code below works.

(App.Current as App).GetRepository()), where GetRepository is the exposed property.

This is OK, in a way, as it is consistent with the fact that all reading and writing of files must be done from the App.xaml file at startup and suspension. Performing any access like this from other pages, causes any asynchrouous file IO to “bomb out” without raising any exceptions. THIS IS NOT GOOD AT ALL.

### Book images

Collect images for the book covers. Name them as the Title, but replacing the spaces with ‘-‘. This will mean there is no need to hold the path in the file..

### Binding to selectable Lists (eg ListView, ComboBox)

The ItemsSource property should be bound to the collection being displayed

The SelectedIndex property should be bound to an index that identifies the current record

The SelectedItem property should be bound to the current record.propertyName, where propertyName represents the property that is to be displayed in the list.

The DisplayMemberPath should also be bound to this propertyName property.

### Design Instance

Investigate the Design Instance and Type of the data context. This seems to make it possible to have temporary data available at design time and therefore allows the pages to “look” realistic as they are defined. Xaml can ignore attributes by designating a prefix as Ignorable. This means that at runtime, the test data can be ignored in preference to the real data store.

# Future Development

## Templates for Survey Questions and Responses

The ability to add template for surveys, question and likert responses would be added to the system. This would facilitate quicker and simpler techniques for setting up surveys..

The current implementation will include the ability to store templates within the Domain Model, and populate these with test data. The maintenance of these templates would be reserved for future development.

This should be reflected in the comments in the About panel, if there is time.