MHoumadi Nourdine

UWP SQLite STARTER KIT

Version 1.0

Contents

[Version 2](#_Toc473133290)

[Description 2](#_Toc473133291)

[Presentation 2](#_Toc473133292)

[Creating a Model 3](#_Toc473133293)

[Creating a View Model 3](#_Toc473133294)

[Creating a View 5](#_Toc473133295)

[Creating services to interact with the SQLite database 7](#_Toc473133296)

[Using services to interact with the SQLite database in the ViewModel 8](#_Toc473133297)

[Displaying datas from the ViewModel to the View 11](#_Toc473133298)

[Executing a command 13](#_Toc473133299)

[Command without parameter 14](#_Toc473133300)

[Asynchronous command with no parameter 15](#_Toc473133301)

[Command with parameter 16](#_Toc473133302)

[Navigation 16](#_Toc473133303)

[Navigation without parameter 16](#_Toc473133304)

[Navigation with parameter 17](#_Toc473133305)

UWP SQLite Starter Kit

# Version

This version is the 1.0

# Description

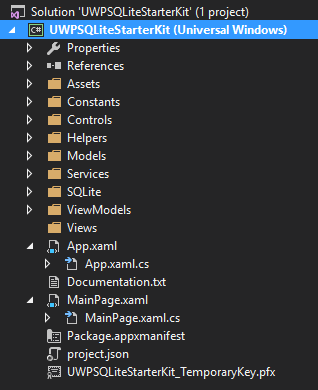
UWP SQLite Starter Kit is a Visual Studio Project template that creates a skeleton of a UWP application using MVVM Light and SQLite database with these features:

* Creating Models will work with SQLite database.
* Creating ViewModels using MVVM Light.
* Creating Views.
* Creating services to interact with the SQLite database.
* Displaying datas from SQLite database using MVVM Light.
* Executing a command.
* Navigating to a page with parameters.

More features coming soon.

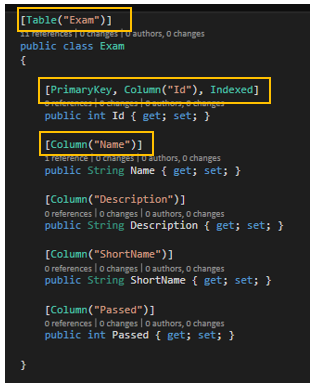
# Presentation

The UWPSQLiteStarterKit, contains the following items:



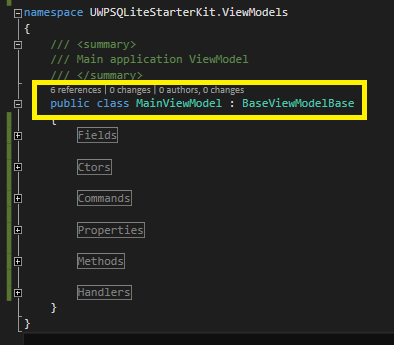
# Creating a Model

Each model need to have SQLite attributs members

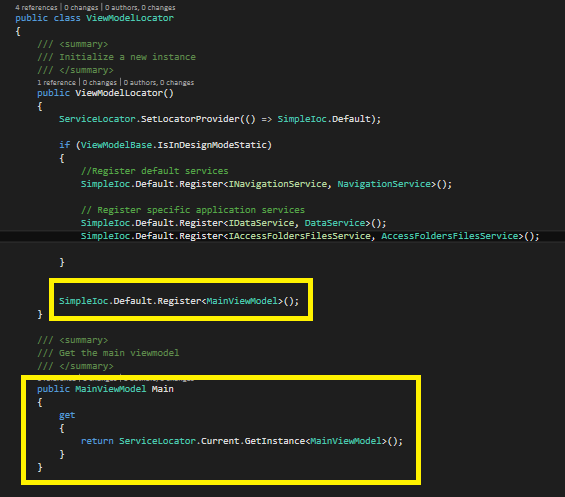


# Creating a View Model

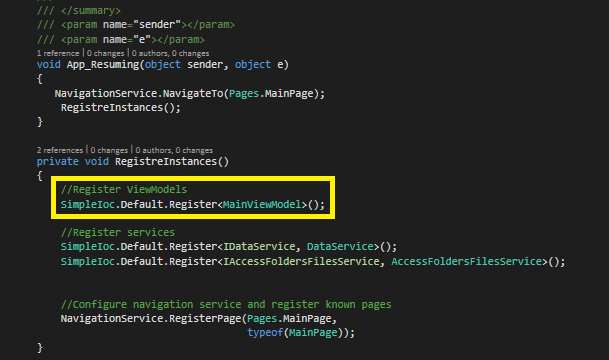
* Each View Model inherits from *BaseViewModelBase*



* To the ViewModelLocator.cs add the “*SimpleIoc.Default.Register<[TheViewModelName]>();*” in the constructor.

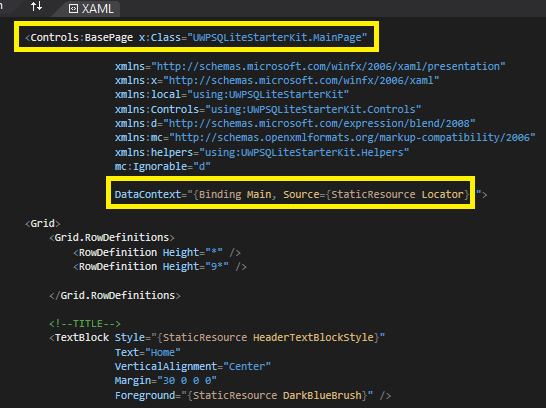


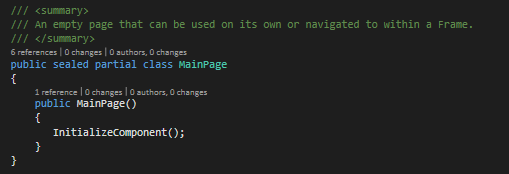
* Add the “MainViewModel” ServiceLocator
* To the App.xaml.cs file in the method RegistreInstances(), register the view model



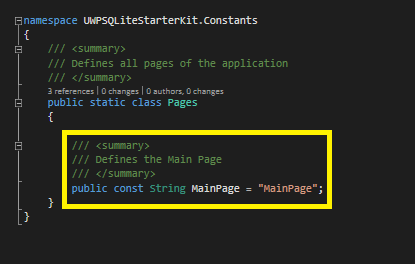
# Creating a View

* Each view inherits from *BasePage* and contains a DataContext which defines in the ViewModelLocator.cs file.

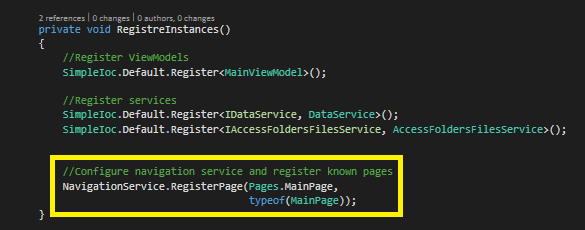




* Add the name of the page in the *Pages.cs* on the repository *Constants*

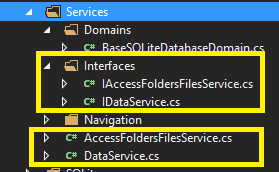


* Configure the navigation service to the page in *App.xaml.cs* on the *RegistreInstances* method

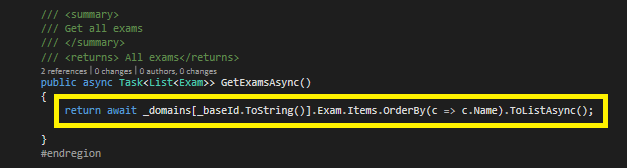


# Creating services to interact with the SQLite database

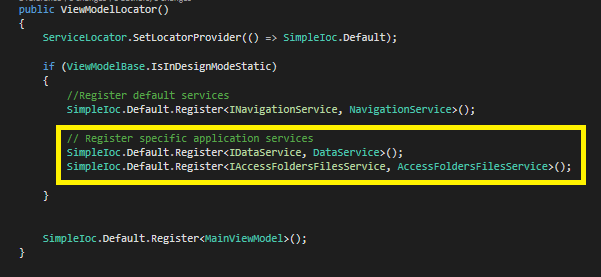
* Creating Interface and Service.



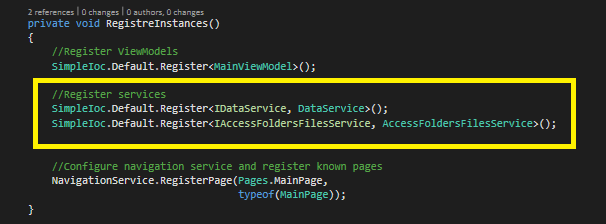
* To retrieve datas from a table, using Async Methods and this syntax : *\_domains[\_baseId.ToString()].TableName.Items.ToListAsync().* Below a sample from the DataService:



* Define services in the ViewModelLocator

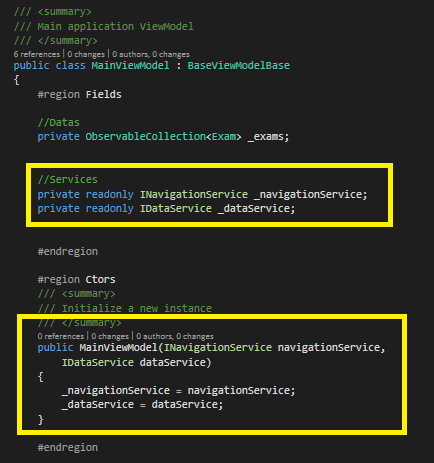


* Define services in the App.xaml.cs

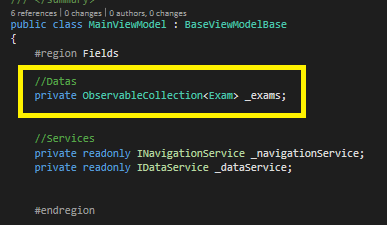


# Using services to interact with the SQLite database in the ViewModel

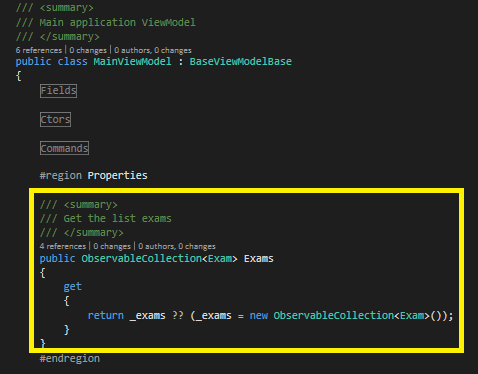
* Declare services in the ViewModel



* Declare Datas (private) to display as *ObservableCollection<T>* in the *Fields* region

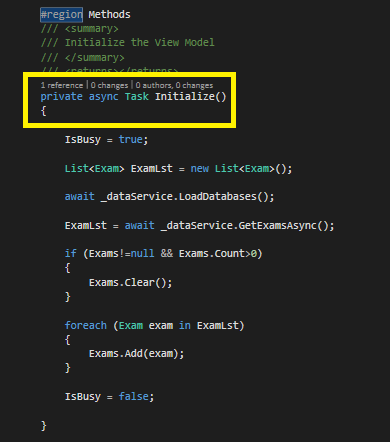


* Declare Datas (public) to display as *ObservableCollection<T>* in the *Properties* region

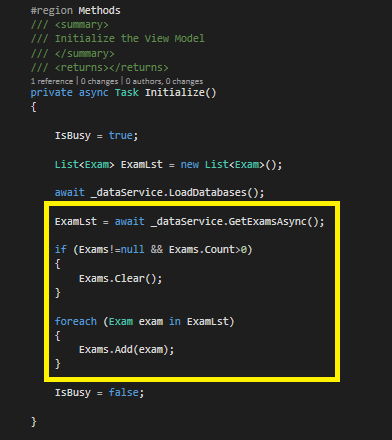


* Create *Initialize()* Task in the *Methods* region

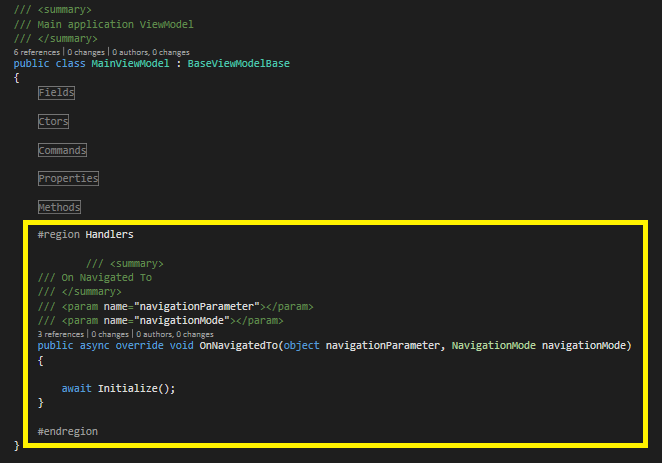
*IsBusy* means that the ViewModel is charging the datas on initialization.



* Retrieve datas from the database using *\_dataService* and insert them to the *ObservableCollection<T>*



* The method *LoadDatabases()* has to be use only once in the first ViewModel of the application. The others ViewModel won’t have this method.
* Create *OnNavigatedTo()* method for using *Initialize()* method



# Displaying datas from the ViewModel to the View

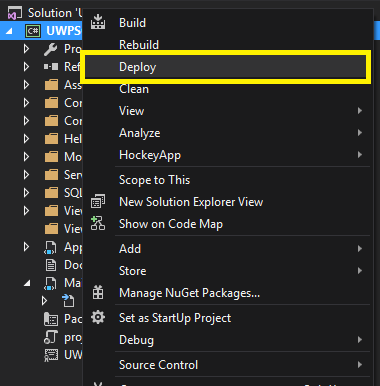
* Create the *DataTemplate* in the *App.xaml*



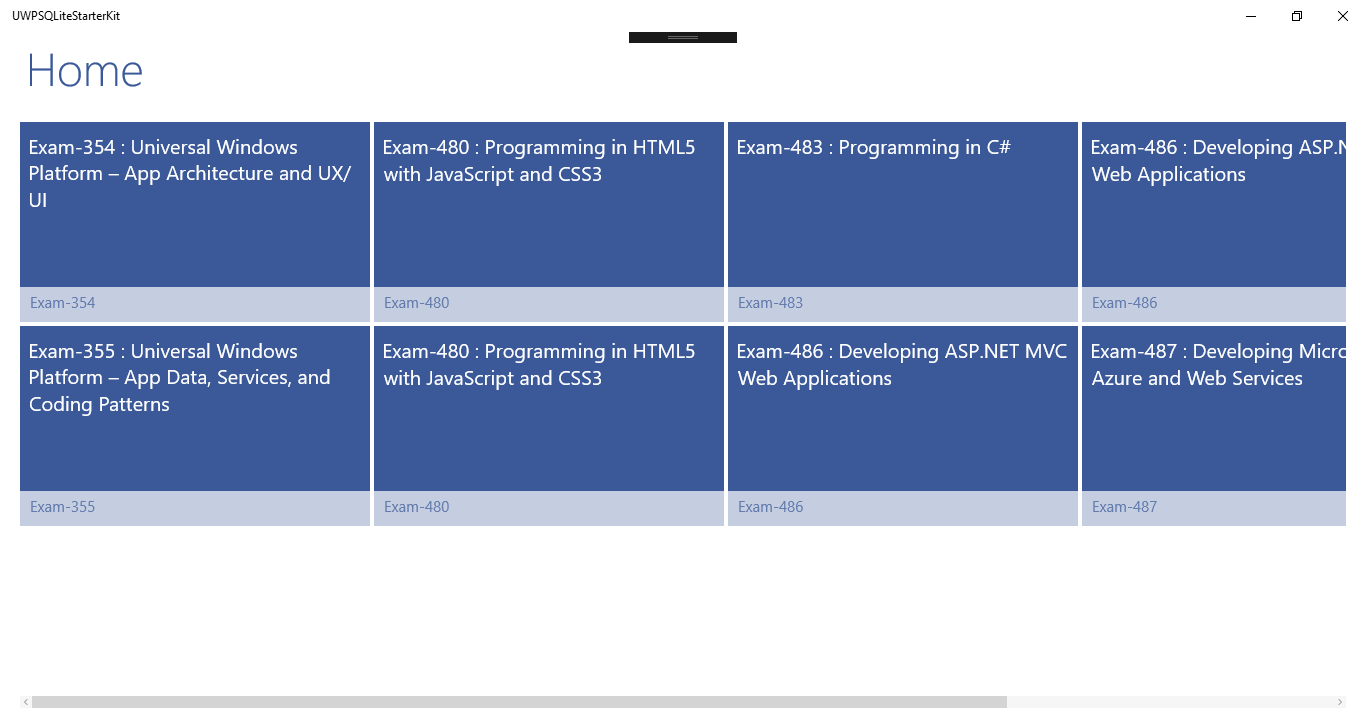
* Displaying the datas (Exams) in the View (MainPage)



* Deploy the project and execute it



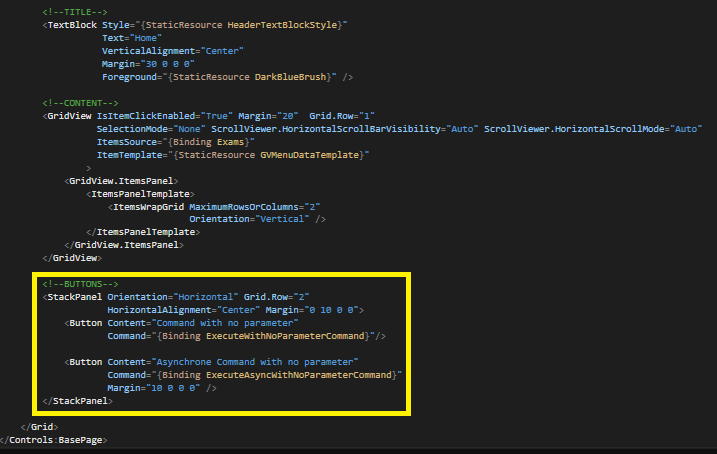
* The HomePage is displaying.



# Executing a command

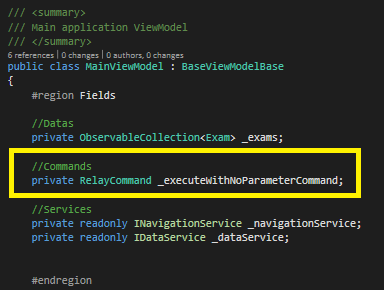
For explaining how to use command in UWP SQLite Starter Kit, I’ve added two buttons:

* Command with no parameter
* Asynchronous Command with no parameter

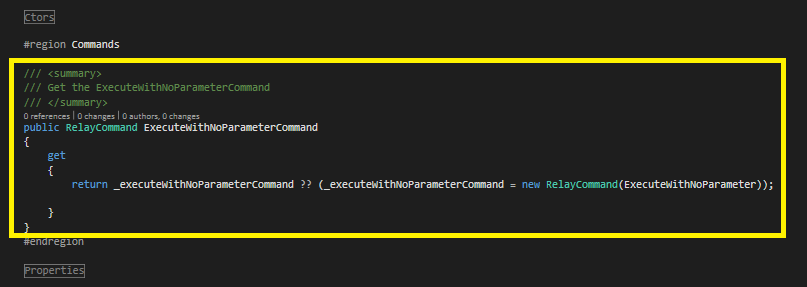


## Command without parameter

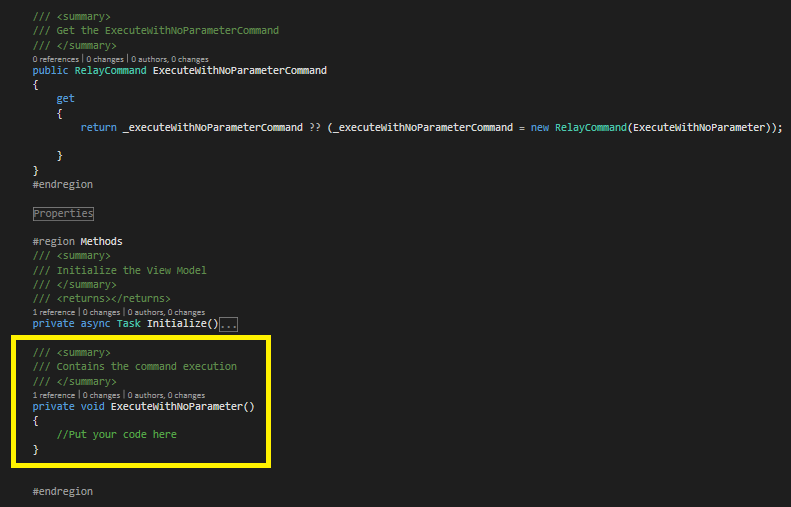
* Declaring command in the *Fields* region on the *MainViewModel.cs*



* Declaring command in the *Commands* region on the *MainViewModel.cs*

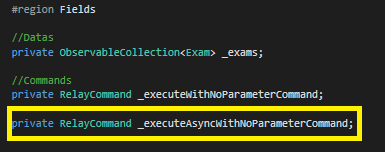


* Creating the *ExecuteWithNoParameter* method

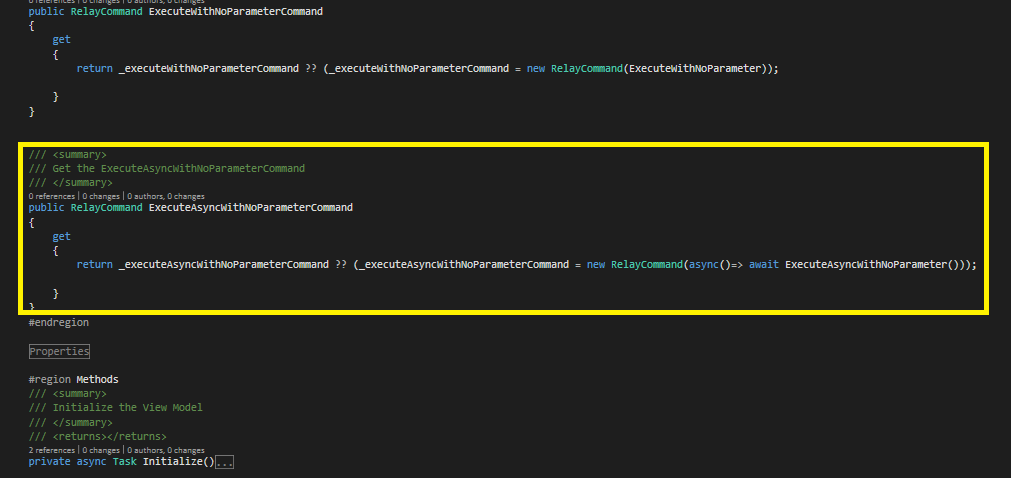


## Asynchronous command with no parameter

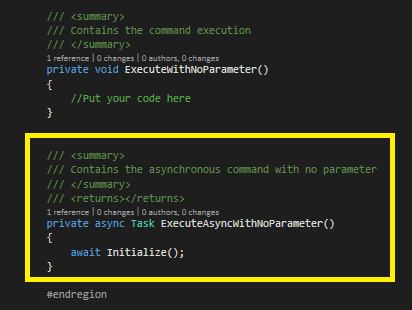
* Declaring command in the *Fields* region on the *MainViewModel.cs*



* Declaring command in the *Commands* region on the *MainViewModel.cs*



* Creating the *ExecuteAsyncWithNoParameter* Task



## Command with parameter

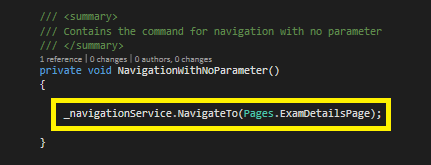
See the next section: Navigation

# Navigation

For explaining how to use Navigation in UWP SQLite Starter Kit, I’ve added one button:

* Navigation with no parameter

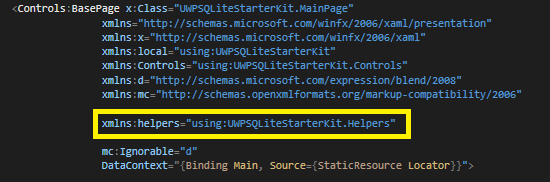
## Navigation without parameter



## Navigation with parameter

For explaining navigation with parameter in UWP SQLite Starter Kit, let’s handle the ItemClick event on GridView in using MVVM.

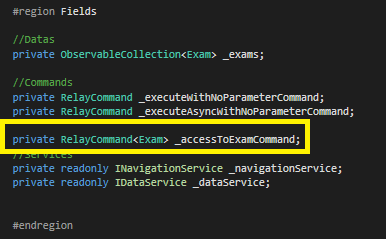
* Add the namespace Helpers in the MainPage.xaml.



* Apply the *ItemClickCommandHelper* to the GridView and bind it to the *AccessToExamCommand*



* Access to the *MainViewModel*.
* Declaring command in the *Fields* region on the *MainViewModel.cs*

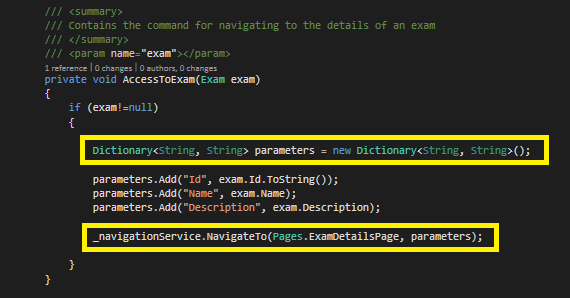


The *ItemClickCommandHelper* get the selected item in the GridView which is an *Exam* type.

* Declaring command in the *Commands* region on the *MainViewModel.cs*



* Creating the *AccessToExam* method



In UWP SQLite Starter Kit, the parameters of the navigation are in *Dictionnary<String, String>* type.