***Testing ConcreteItemDAO***

The intro to this chapter was written by Laura when testing the OrderDAO.

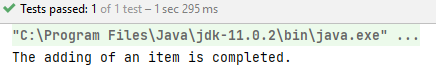
**createItem(Item item)**

* Should never throw anything, insert the Item object into the database.
* It will never be called with a null object therefore it is unnecessary to check it.

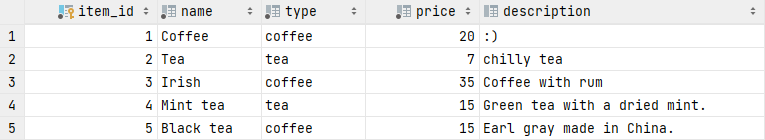
*JUnit test method*

******

*JUnit output*

******

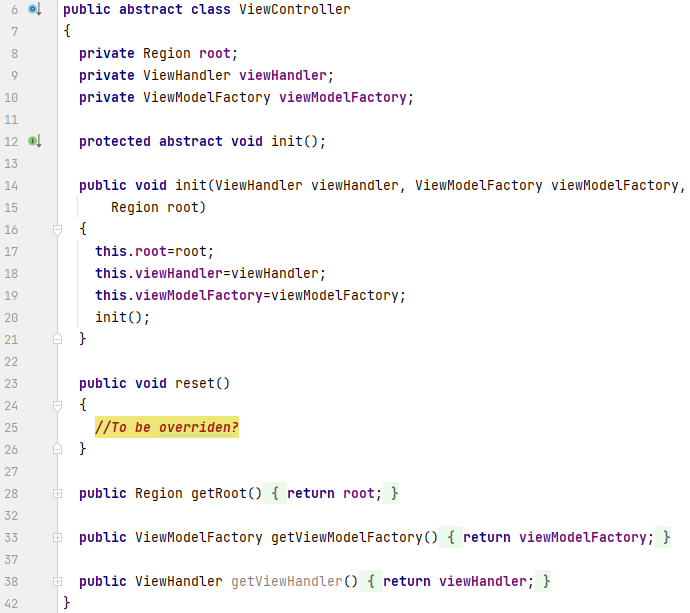
*The changes in the database*

******

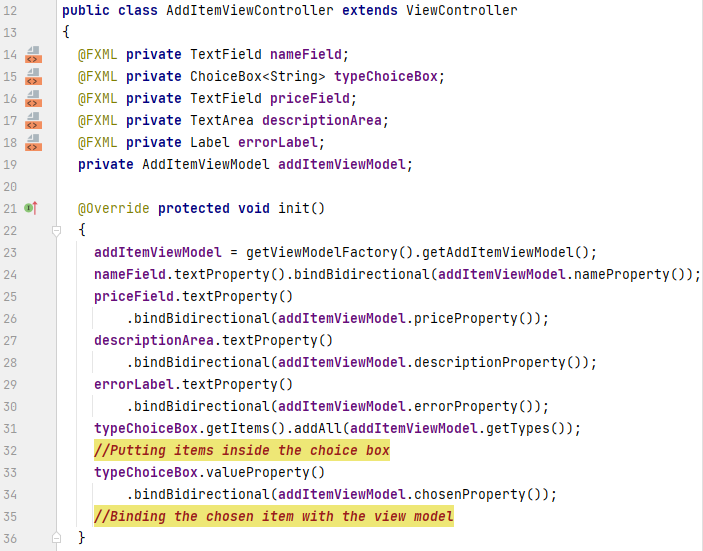
***Documentation of AddItemViewController***

Each view requires a ViewController so that it can be connected with the functionality of the system. All of the specific ViewControllers extend the superclass ViewController, which is simply responsible for initializing ViewHandler, ViewModelFactory and root. There are also getters for all of the fields. The important part of this abstract class is that it has a *protected abstract void init()* method. This method is essential for the specific ViewController as this is where all the necessary initializations are occurring.

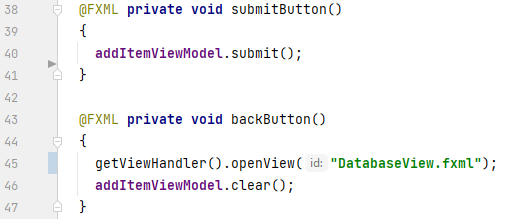
*ViewController superclass*

**

*Fields and init method*

  
  
The mentioned *init()* method initializes the fields of the class and binds them with the StringProperty inside the AddItemViewModel class. This ViewModel’s value is assigned by calling a *getViewModelFactory()* from the superclass. After this the AddItemViewModel is retrieved by a simple getter. Line 31 is also worth mentioning as this is the way to fill the choice box with the determined values. This is done by getting the ArrayList of types from AddItemViewModel. Moreover, method *valueProperty()* is used to obtain the value chosen by the user from the choice box. Subsequently, it is bound with the corresponding StringProperty inside the AddItemViewModel.

*OnAction methods*



The method *submitButton()* is delegating the work to the AddItemViewModel and all the information inside the ViewModel is fetched from the bound values. The method *backButton()* solely opens the previous View and calls a *clear* method inside the AddItemViewModel.

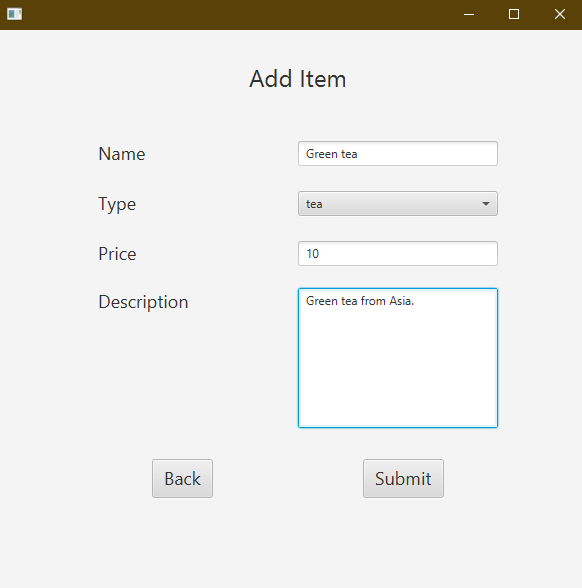
# StartView, LoginView, DatabaseView Documentation

The initial view presented to the users is the start view. Two choices are granted: the first one is to start ordering; this would open the CustomerView where the items can be added to the order, while the second option is to log in: this would open the LoginView where the employees can enter their access key in order to be granted permission to see the DatabaseView. In this version they can select Add Item, which will take them to the AddItemView where they can add new items to the database.

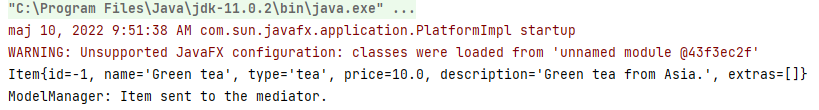
***Documentation of AddItem integration testing***

For the purpose of testing the Add Item case the integration test was organized. This will test if the information is being transferred correctly from the View up to the database going through ViewModel, Model, RMI connection and the Database Persistence.

*AddItemView*

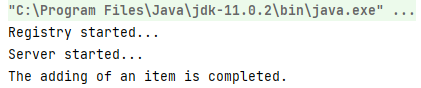


*Console output on the Client side*

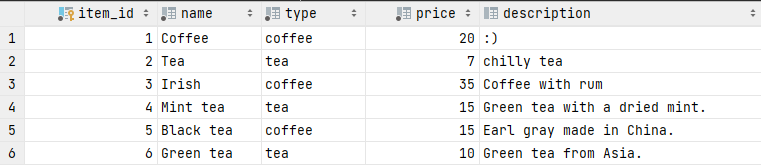


As it can be seen, the Item object is created properly from the given values and then sent to the mediator package which is tested by the output in the console.

*Console output on the Server side*



*The item in the database*



Adding of an item is completed without any errors which is checked by the proper console output and the existence of the specific item in the database.