Online Market Place Application

Assignment #2: Report

Implementation using Front Controller Pattern, Command Pattern and Abstract Factory Pattern

Course: CSCI 50700 - Object-Oriented Design and Programming

By
Mani Manjusha Kottala
IUPUI, Computer Information Science

Table of Contents

Introduction:

In assignment 1, an application is developed using (MVC) Model-View-Controller and Java RMI. For this assignment, the main goal is to achieve the login functionality for user, admin and to use Front Controller pattern, Command pattern and Abstract Factory pattern.

Assignment 1 – Fix:

In assignment1, I haven't created clear separation between the view logic and frame work functionality. To address the issue, I have added a controller on the client side for establishing the RMI connection to the server. MarketPlaceClientController class is used for interaction between the server-side controller and the client views.

Front Controller Pattern:

Main aim of Front Controller pattern is to provide a centralized request handling mechanism. In online market place application, different views are to be implemented for User(customer) and Admin. It mainly has 3 components[1]-

1.FrontController: Handler for handling requests of the application,

2. Dispatcher: Dispatcher is responsible for redirecting to respective view

3. ClientEntryView: This is where either admin or user enters his details.

In the first step, either admin or user enters his/her user id and password into terminal. Based on the type of account selected, MarketPlaceClientController Sends request to the FrontController. FrontController checks for the user/admin authentication by interacting with MaketPlaceClientController, a remote method call is made to the MarketPlaceController (server-side controller) for verifying the user credentials. After successful authentication, FrontController forwards the request for view to Dispatcher. This Dispatcher class is responsible for generating either admin view or user view based on the request type.

Abstract Factory Pattern:

This pattern is used to create multiple factories which further can create products, in this case it would be object creation. In this assignment, this pattern is used to create factor of views for admin and user. A factory of related objects can be created using a common interface using this pattern. Below is the list of classes that are involved in building abstract factory pattern[4]:

- FactoryProducer: FactoryProducer is used by Dispatcher to create factories using admin or user information
- AbstractFactory: This is an abstract class which can help producer to get admin and user objects
- AdminFactory: This class inherits from AbstractFactory to generate concrete objects of admin type
- UserFactory: This class inherits from AbstractFactory to generate concrete objects of user type
- Admin interface and User interface: These interfaces provide an abstract method which can be overridden by implementing classes
- MarketPlaceUserView and MarketPlaceAdminView: Both these views are concrete classes which are implemented using above two interfaces respectively.

Command Pattern:

Command pattern is helpful in dealing with situations where an event can be fired at a later time by an object which stores information related to methods and respective object owner. To design this pattern, one should implement invoker, command, receiver and client. Actions class acts as command in this application which has a execute() method responsible for executing the recevied commands. But only Invoker class decides which commands to execute using the command interface. Also, Invoker is unaware of the what a concrete command is. Dispatcher acts as client and MarketPlaceAdminView is the receiver. Actions command interfaces implements four classes AddItems, DeleteItems, BrowseItems and UpdateItems. Based on the type of object created, Invoker executes its respective action.

Domain Model [2] and Class Diagram [3]:

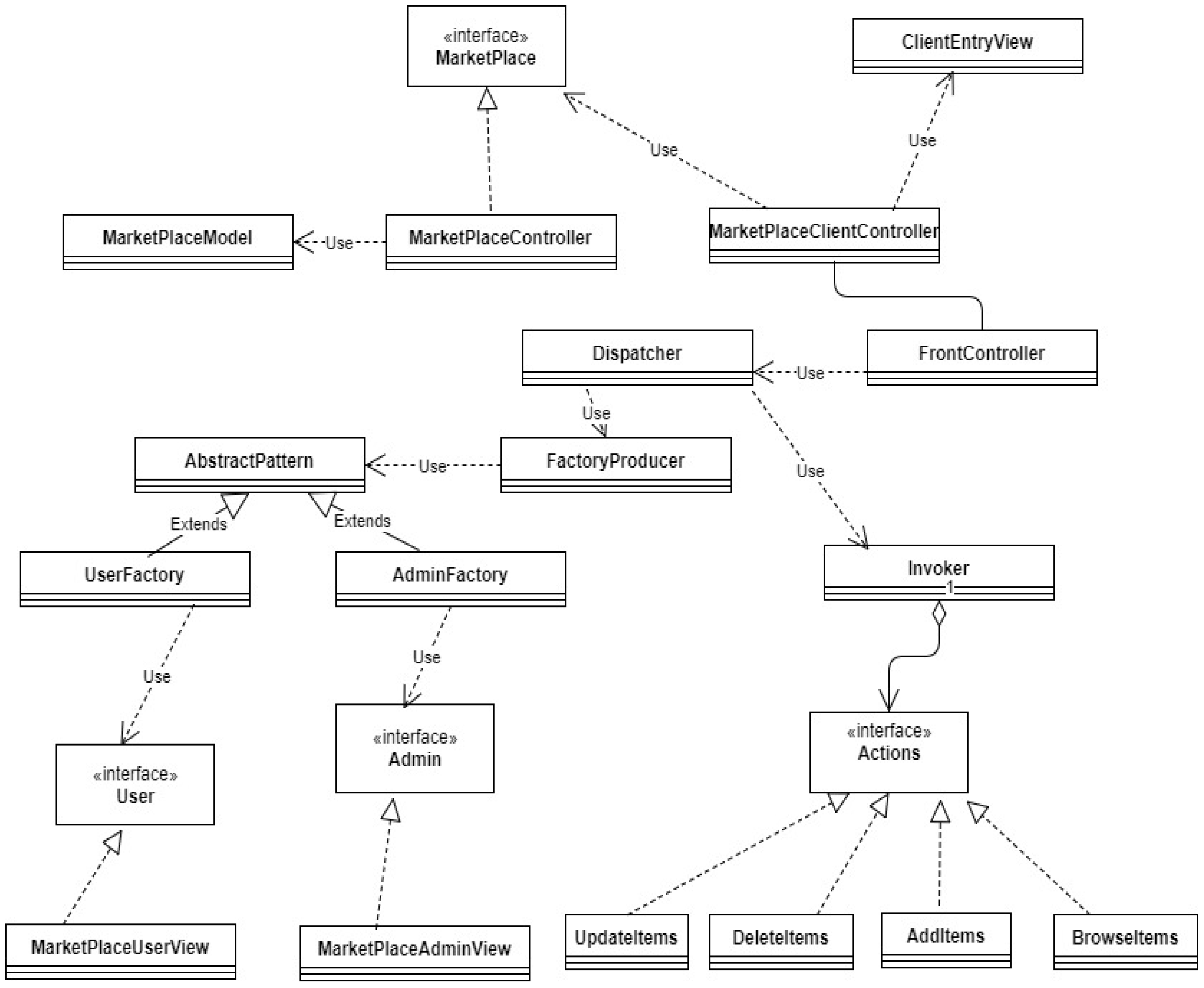


Fig 1: Domain model for MarketPlace Application

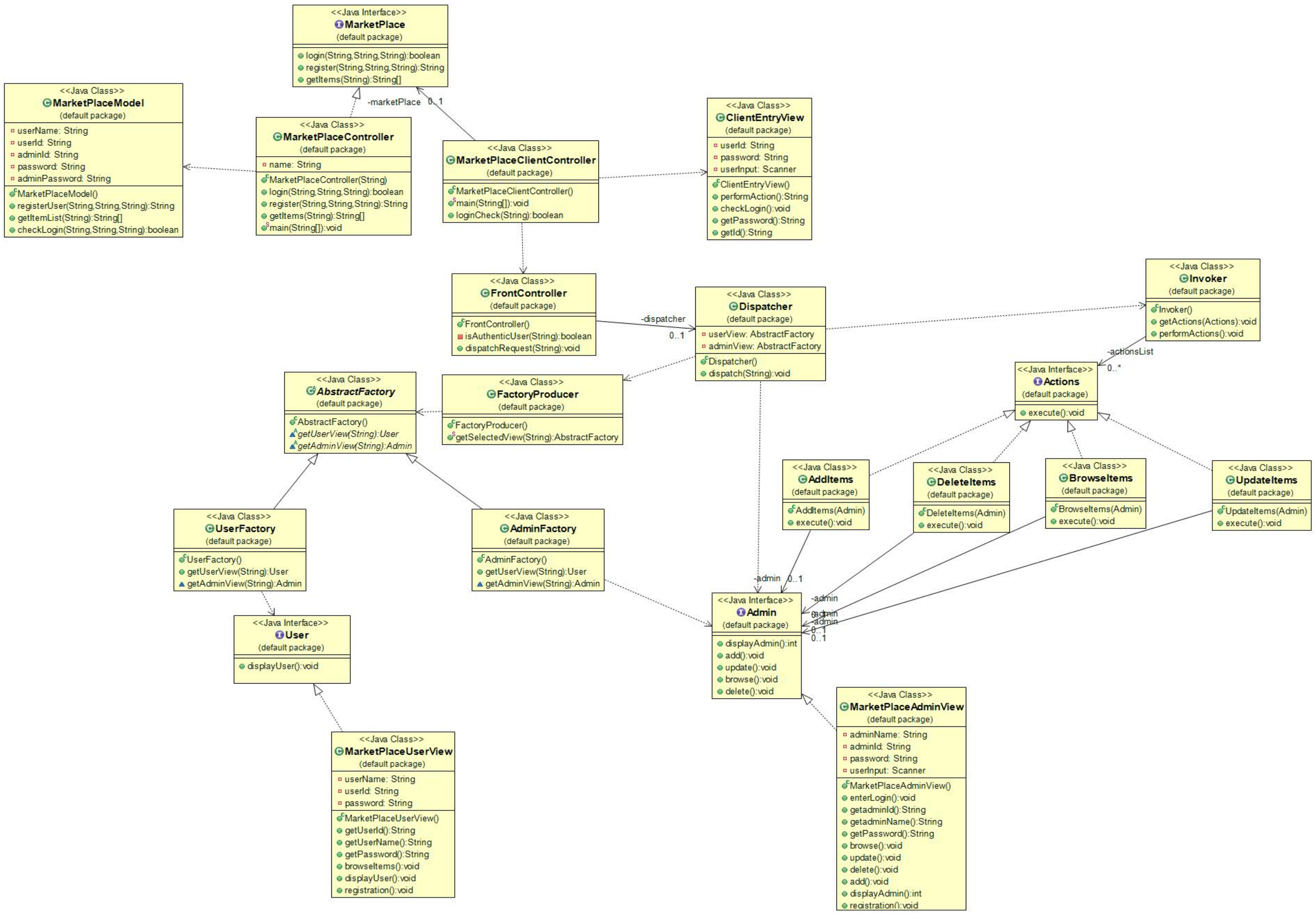
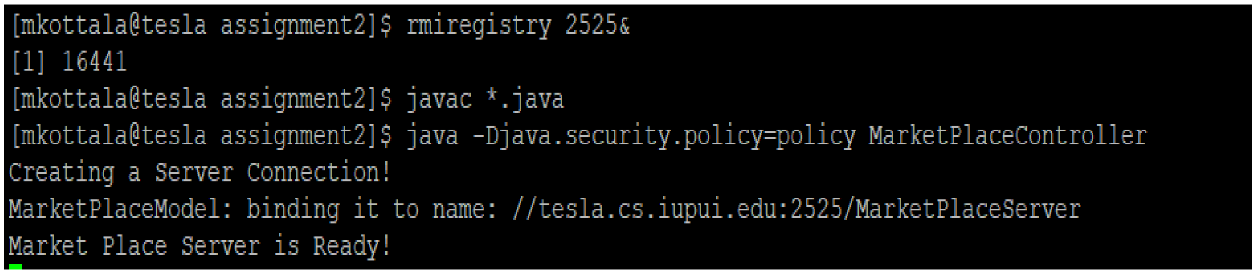


Fig 2: Class Diagram for MarketPlace Application

Sample runs:

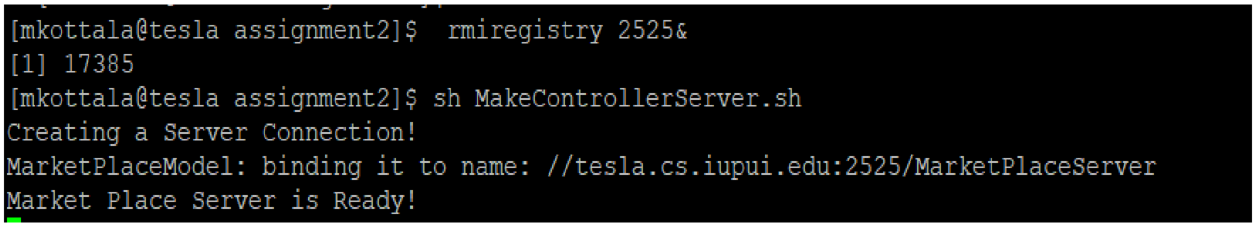
1. Starting the RMI registry and Running the server application



```
[mkottala@tesla assignment2]$ rmiregistry 2525&
[1] 16441
[mkottala@tesla assignment2]$ javac *.java
[mkottala@tesla assignment2]$ java -Djava.security.policy=policy MarketPlaceController
Creating a Server Connection!
MarketPlaceModel: binding it to name: //tesla.cs.iupui.edu:2525/MarketPlaceServer
Market Place Server is Ready!
```

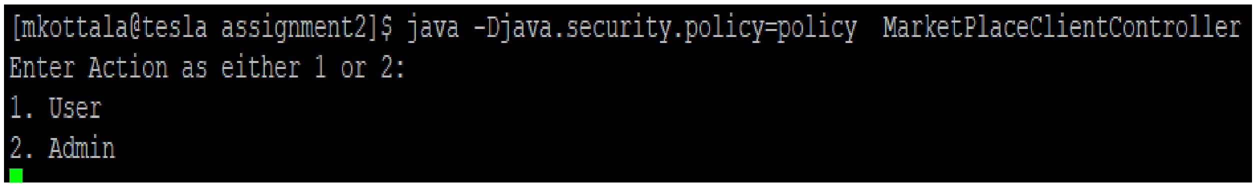
Or

Using makefile: MakeControllerServer.sh



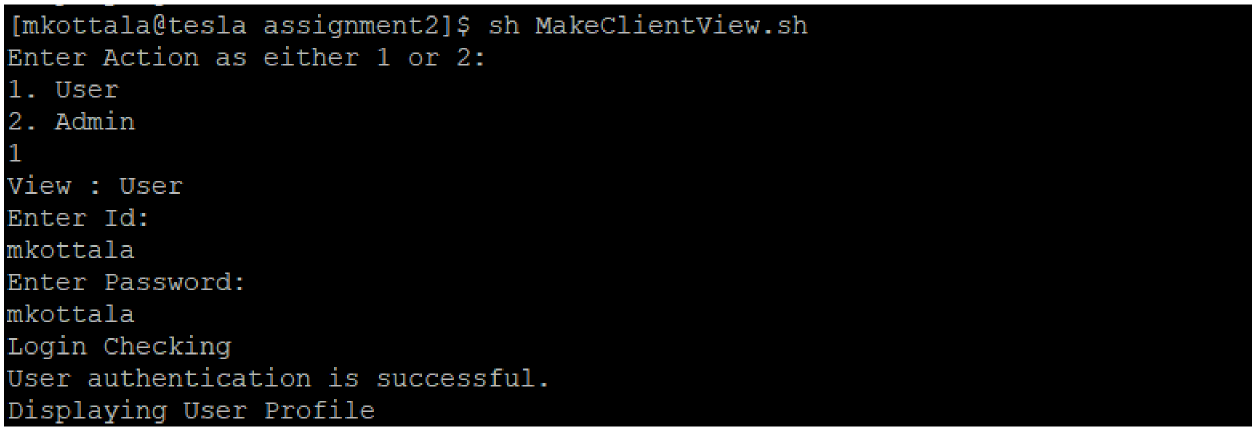
```
[mkottala@tesla assignment2]$  rmiregistry 2525&
[1] 17385
[mkottala@tesla assignment2]$ sh MakeControllerServer.sh
Creating a Server Connection!
MarketPlaceModel: binding it to name: //tesla.cs.iupui.edu:2525/MarketPlaceServer
Market Place Server is Ready!
```

2. Running the Client application

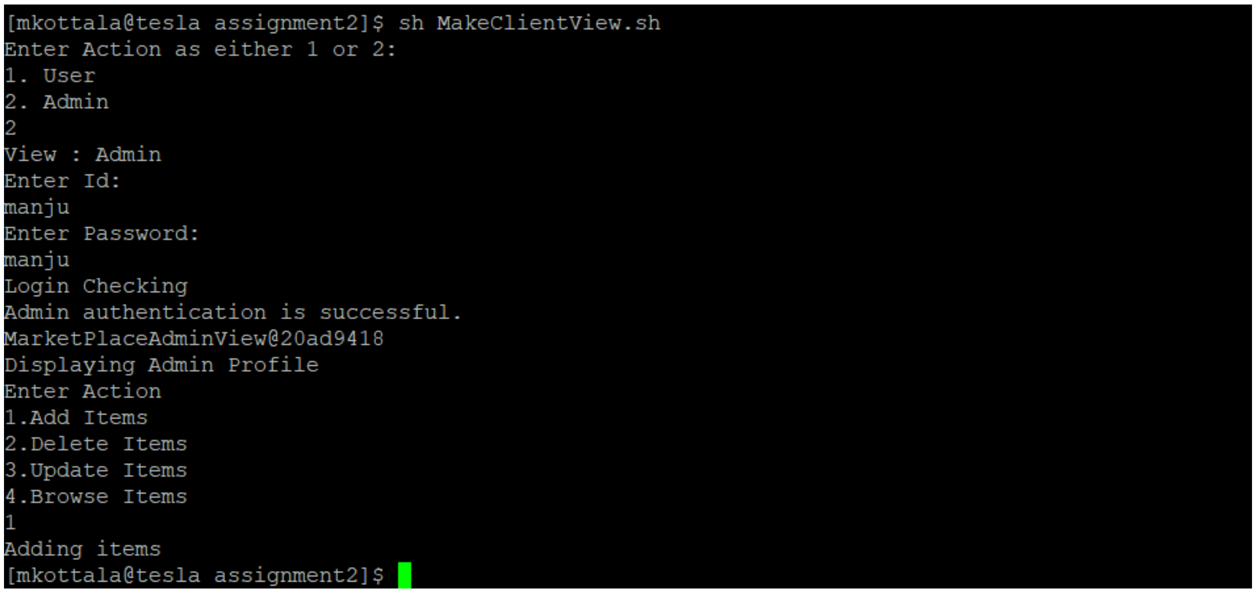


```
[mkottala@tesla assignment2]$ java -Djava.security.policy=policy  MarketPlaceClientController
Enter Action as either 1 or 2:
1. User
2. Admin
```

Or
Using MakeFile: MakeClientView.sh

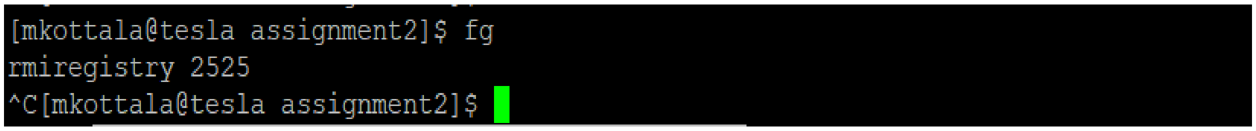


```
[mkottala@tesla assignment2]$ sh MakeClientView.sh
Enter Action as either 1 or 2:
1. User
2. Admin
1
View : User
Enter Id:
mkottala
Enter Password:
mkottala
Login Checking
User authentication is successful.
Displaying User Profile
```



```
[mkottala@tesla assignment2]$ sh MakeClientView.sh
Enter Action as either 1 or 2:
1. User
2. Admin
2
View : Admin
Enter Id:
manju
Enter Password:
manju
Login Checking
Admin authentication is successful.
MarketPlaceAdminView@20ad9418
Displaying Admin Profile
Enter Action
1.Add Items
2.Delete Items
3.Update Items
4.Browse Items
1
Adding items
[mkottala@tesla assignment2]$
```

3. Terminating RMI registry:



```
[mkottala@tesla assignment2]$ fg
rmiregistry 2525
^C[mkottala@tesla assignment2]$
```

Conclusion:

In this assignment, I have gained knowledge on implementation of MVC pattern, Front Controller pattern, Abstract Factory pattern and Command pattern. I have realized the faults in assignment-1 and tried to fix those in this assignment. I have also understood the usage of different patterns for creating an application.

References:

[1] https://www.tutorialspoint.com/design_pattern/front_controller_pattern.htm

[2] https://en.wikipedia.org/wiki/Domain_model

[3] https://dzone.com/articles/uml2-class-diagram-java

[4] Examples discussed in class