

---

# ONLINE MARKETPLACE

---

Pritesh Ratnappagol



JANUARY 30, 2018

INDIANA UNIVERSITY – PURDUE UNIVERSITY INDIANAPOLIS

## Table of Contents

<b>Introduction :</b>	<b>3</b>
<i>Remote method invocation (RMI) :</i>	<i>3</i>
<i>Model-View-Controller (MVC)</i>	<i>3</i>
<b>Assignment Discussion :</b>	<b>4</b>
<b>Outline of the classes used for project:</b>	<b>5</b>
<b>Domain Model:</b>	<b>6</b>
<b>Working of the code and sample runs :</b>	<b>7</b>
<b>Screenshots:</b>	<b>8</b>
<b>Conclusions :</b>	<b>8</b>
<b>References :</b>	<b>8</b>

## Introduction :

### Remote method invocation (RMI) :

- Remote method invocation (RMI) is an API, which implements RPC in java with support of object oriented paradigms.
- RMI operates only in Java domain.

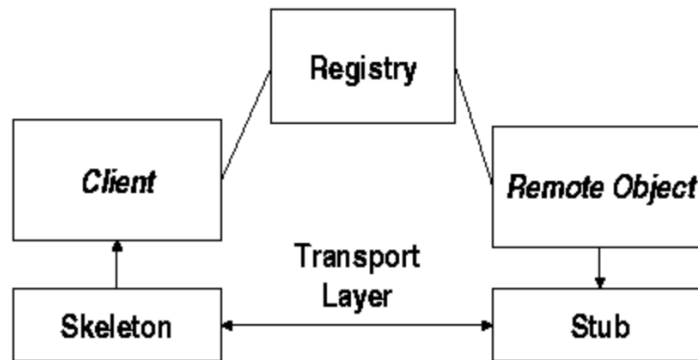


Fig 1. RMI Architecture.

- RMI is a client-server model.
- The stub acts as a proxy for the remote object.
- The skeleton is an object that lives in the same JVM as the remote object and handles communication with the stub.
- The registry is used to manage remote references.
- The server binds a remote reference to itself on to the registry.
- To obtain a remote reference to the server, the client contacts a registry which may be on a different remote host.
- The client can use the remote reference obtained from the registry to invoke methods on the remote object.

### Model-View-Controller (MVC)

MVC is a Software Architectural Design Pattern. The main of the MVC Design Pattern is to separate the logic (model) from interface (view). This helps us to promote a organized programming and allows multiple developers to work on the same project.

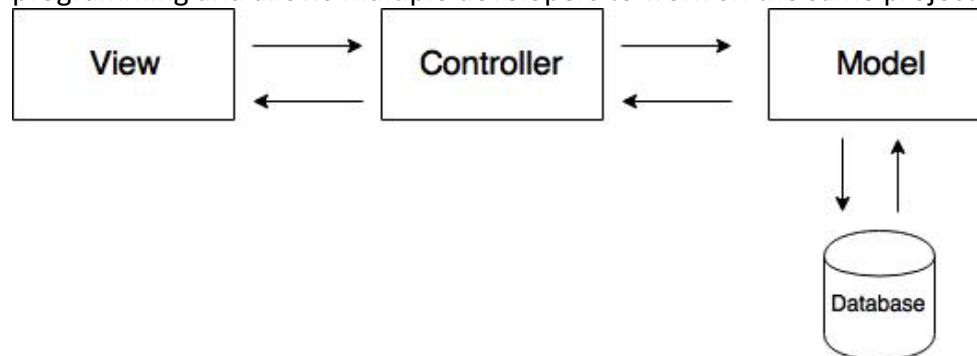


Fig 2. Model-View-Controller

It has 3 components in it – Model, View and Controller.

Model :

- Model is responsible for getting the data related logic and usually it interacts with some kind of database like mySql.
- Model communicates with the controller.
- The controller can request the data through the model and in most cases controller updates the view and in some cases model directly updates the view.

View :

- Actual view of the application is the user interface.
- It is what the user see's when he/she interact with the application.

Controller:

- It takes the user input from the view.
- Controller acts as the middleman between model and view.
- It gets the data from model.
- Passes the data to the view.

### Assignment Discussion :

In order to develop Online Marketplace Application we are going to use the above RMI technology explained for the client and server program which will be developed using the MVC architectural pattern. In this, model will act as server side, and view and controller will act as client side.

The proposed system will consist of 3 sections:

Administrator:

Administrator will be able perform these actions –

- 1) Adding new admin
- 2) Browse Items
- 3) Update Items
- 4) Remove Items
- 5) Add items
- 6) Add/Remove Customers
- 7) Review/Change the order details

Customer :

- 1) Browse Items
- 2) Add items in the shopping cart
- 3) Place order

Guest :

- 1) Browse Items
- 2) Get registered

## Outline of the classes used for project:

- 1) User :  
It will consist of the username and password which will be used to login in the application by Customer and Administrator.
- 2) Customer :  
Customer will be the one how will be already registered to the application and will consist of attributes such as Name, Address, email, contact, Payment Info and can update any information from the attribute as needed and browse the items and add them in the shopping cart and finally order those selected item from the cart.
- 3) Administrator:  
Administrator will be the one how will be managing the working of the application such adding new administrators or employee's or adding/ removing the customers and the items by updating, removing and adding items and make possible changes in the orders of the customer as per customer requirements.
- 4) Guest:  
Guest can view the items on the application without getting registered and also can register if they want to be customer.
- 5) Items:  
Item class consists of attributes such as unique item-id, type, description and price which can be browsed by customer, Administrator and guest.
- 6) Shopping Cart :  
Shopping cart will consist of items that are added by the customers with attributes such as unique Cart-id , Item-id, Quantity of the item, and estimate total price for the items in the cart.
- 7) Stock:  
Stock class will consist of item-id and availability of the items in the term of quantity. It will be used to check the whether the quantity of the particular item is available as per the customer requirements from the shopping cart and if the stock does not have the desired quantity it will display the error message.
- 8) Order:  
Order is the class where customer finally orders the items in the cart and it will have attributes Order-id, cart-id, date, customer name, total price and shipping-id.
- 9) Order Details  
Order details will consist of the items the customer has purchased. It will be kind of the confirmation receipt of the items customer has purchased. It will consist of the attributes such as order-id, item-id, quantity, per unit price of item and total amount paid for items.
- 10) Shipping  
Shipping class will consist of shipping-id, type of the shipping (e.g. – next day shipping or normal shipping) and the cost of the shipping.

## Domain Model:

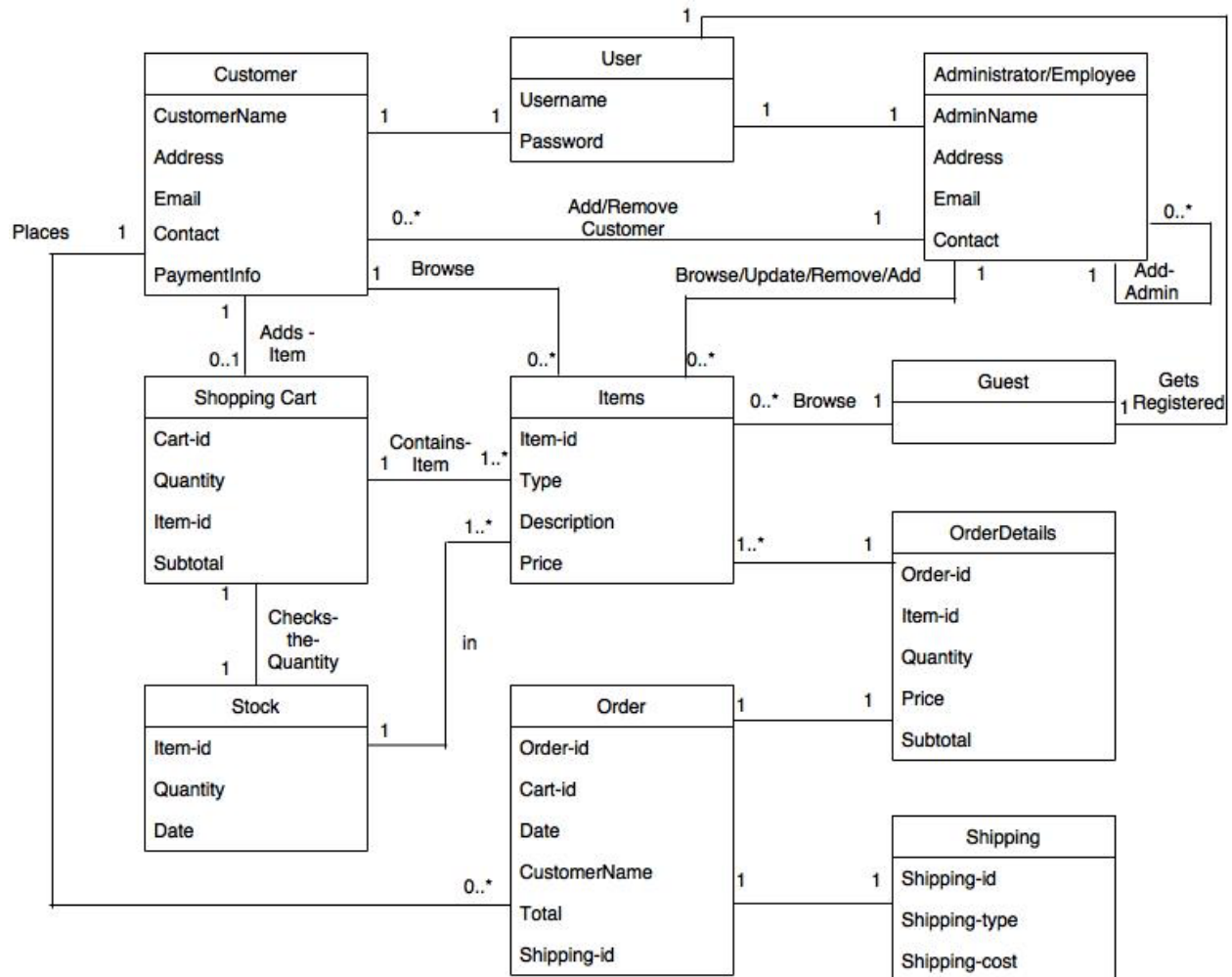


fig 3. Domain Model for online Market Place

The domain model in the fig.3 consist of the classes explained above with required associations and attributes needed for the project.

## Working of the code and sample runs :

In order to understand the working of the Java RMI using MVC pattern I have developed a simple code where user which is at the client side enters his username and sends the request to the server and server responses the client by giving him the greeting message along with his username.

To accomplish this using MVC pattern I have divided the RMI code of the client server, where server will consist of the model part and RMI interface, and client part will have view and controller.

Actual working:

- 1) View part asks the client to enter the username.
- 2) When client enters the username it sends it to the controller.
- 3) Controller then forwards it to the model where processing is done.

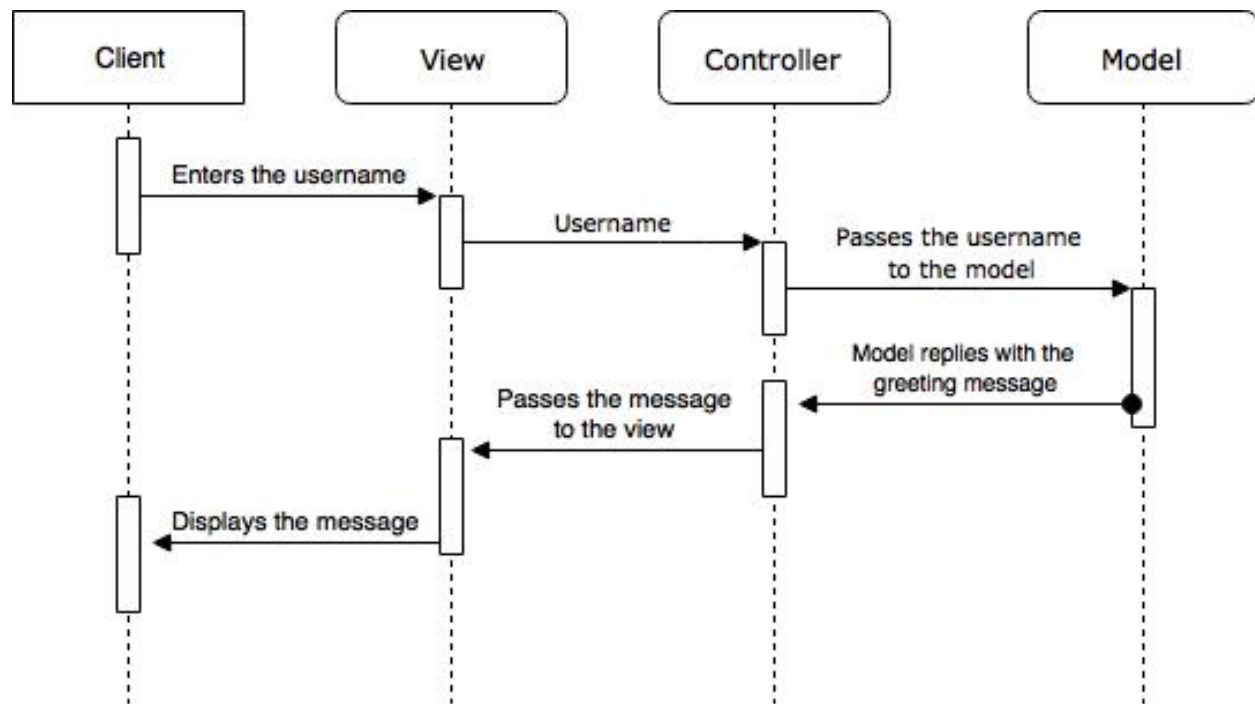
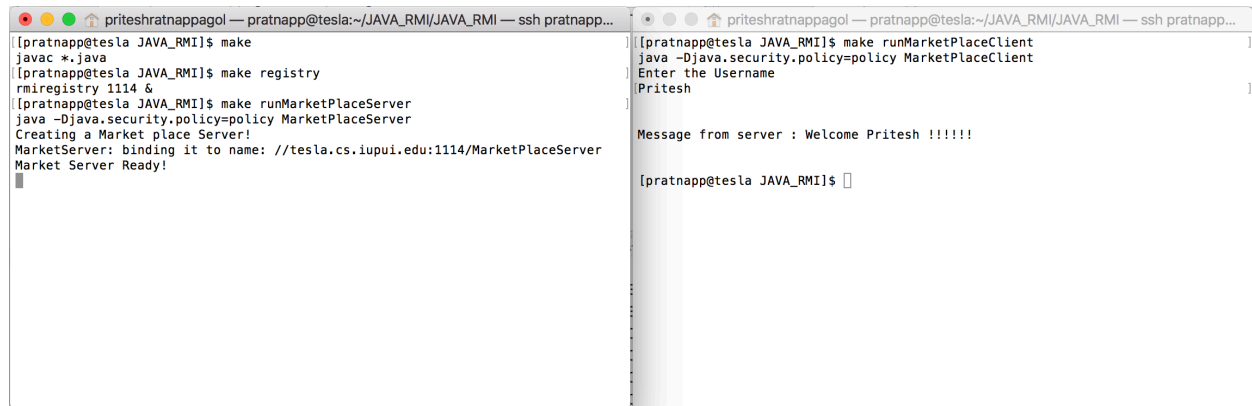


Fig 4. Sequence Diagram

Fig. 4 shows the sequence diagram for the implemented RMI program.

- 4) In model when it receives the username form controller it adds the greeting message to the username and forwards it to the controller.
- 5) Then controller forwards it the view part by calling the function in the view and displays the greeting message to the client

## Screenshots:



```
priteshratnappagol — prtnapp@tesla:~/JAVA_RMI/JAVA_RMI — ssh prtnapp...
[prtnapp@tesla JAVA_RMI]$ make
javac *.java
[prtnapp@tesla JAVA_RMI]$ make registry
rmiregistry 1114 &
[prtnapp@tesla JAVA_RMI]$ make runMarketPlaceServer
java -Djava.security.policy=policy MarketPlaceServer
Creating a Market place Server!
MarketServer: binding it to name: //tesla.cs.iupui.edu:1114/MarketPlaceServer
Market Server Ready!

priteshratnappagol — prtnapp@tesla:~/JAVA_RMI/JAVA_RMI — ssh prtnapp...
[prtnapp@tesla JAVA_RMI]$ make runMarketPlaceClient
java -Djava.security.policy=policy MarketPlaceClient
Enter the Username
Pritesh

Message from server : Welcome Pritesh !!!!!

[prtnapp@tesla JAVA_RMI]$
```

Screenshot 1. Illustrating the implemented system

The above screenshot shows the working of the implemented program. The left terminal is of the server and right terminal is of the Client. User inputs the username on the client side and gets the greeting message from the server.

## Conclusions :

In terms of implementing the RMI in MVC pattern I realized that it makes the programming part easier for implementing the rest of the project as we will be separating the user interface part from the logic so that we can work independently on each side and then make the things work together easily by calling the functions in the controller part. And for the Assignment requirements of the online market place application I have covered all of the parts as the client's requirements but will update the model in the future if any changes are needed.

## References :

- 1] Design of an XML based Interoperable RMI System: SoapRMI C++/Java 1.1.  
[http://www.extreme.indiana.edu/xgws/papers/soaprmi\\_design/index.html](http://www.extreme.indiana.edu/xgws/papers/soaprmi_design/index.html)
- 2] Model View Controller <https://en.wikipedia.org/wiki/Model-view-controller>.
- 3] UML [https://en.wikipedia.org/wiki/Unified\\_Modeling\\_Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language).