Assignment #2 Report

Russell Sherman

February 16, 2017

CSCI 50700

Requirements for the Online Marketplace

After meeting with the client, it was determined that they would like an online marketplace built that would allow them to sell good and possibly services to users all over the world. They would like the ability to separate what the customer can see, and what an administrator of the marketplace would see. Customers should have to register for an account upon entering the marketplace. After registering, the customer would then be required to login to enter the marketplace. The customer should be able to browse the marketplace and see a number of items available for purchase. The items should be assigned a ‘type’, ‘description’, and ‘price’. Customers should be able to add these items to a shopping cart that is specific to the user. The items will have a supply amount, restricting customers from purchasing more than is available. The customers shopping cart should keep track of items added or deleted from the cart and be persistent throughout the customers interactions in the marketplace. Administrators will have the ability to update descriptions, prices, and quantities of the items. They will also be able to remove items from the marketplace. Administrators should have the ability to add other administrators and also add or remove customers from the marketplace. Administrators should not have the ability to purchase items as an administrator; they would have to do so as a customer. The marketplace system should be reliable and be able to handle multiple requests during execution. The system should be able to handle any scenario gracefully.

Requirements for Assignment #2

As we continue to build our Online Marketplace application, the next step was to refine our Model-View-Controller architecture to include an Application Control Pattern. This included implementing a Front Controller that would allow client requests to be channeled through the Front Controller in order to transform the requests into more specific requests. For our purposes in this assignment, the Front Controller will server on the client side and be the middle man between the model and views.

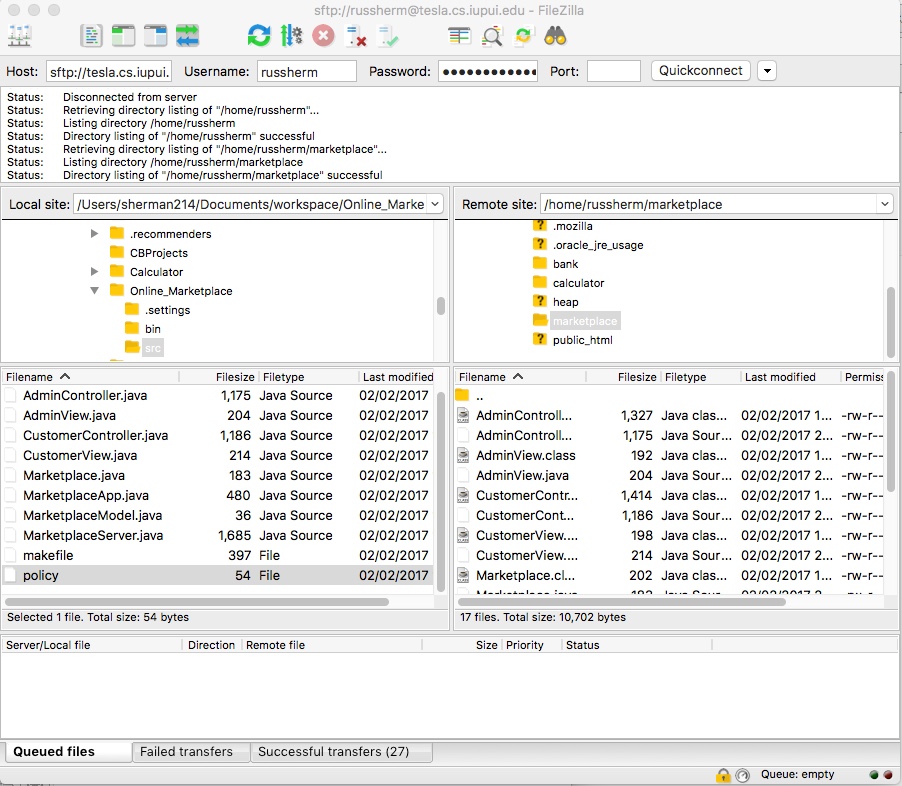
The other addition for assignment 2 was to implement some sort of login authorization. This would include login for both the customers and administrators. I implemented Command and Abstract Factory Patterns to begin to partially implement the login requirement. In order to achieve this, I made “Ryan” as the administrator and everyone else a customer. I will continue to expand this function in later assignments.

Changes Made

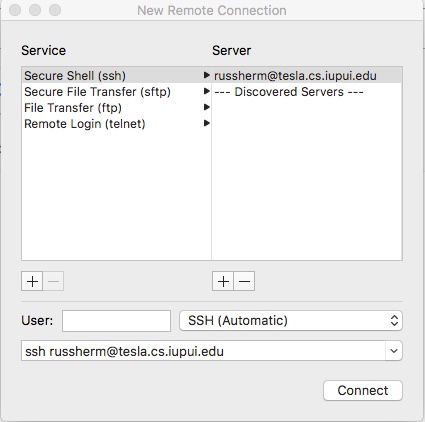
To complete assignment 2 I had to make multiple changes to my code from assignment 1 and also add a number of files. To begin, I got rid of the Marketplace Server file and implemented everything for the server in my Model file. This allowed me to follow a more true to form Model-View-Controller architecture. I also added the Front Controller file that would give the application the ability to delegate responsibilities. Another aspect of the Front Controller was the Dispatcher, which will help to dispatch to the specific view. In order to implement the Abstract Factory Pattern, an AbstractFactory and UsersFactory class would need to implemented. This will allow the creation of Users, whether Customers or Administrators

Sample Run

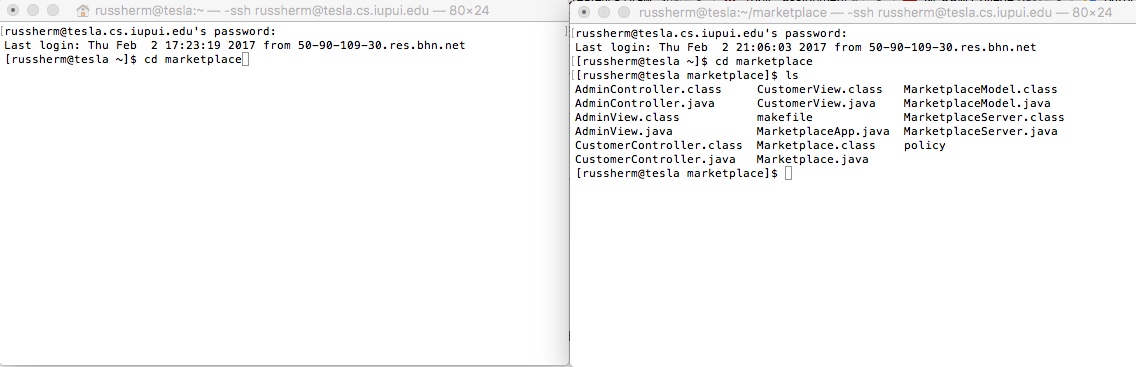
After saving my files in Eclipse, I open a program to transfer my files from my local machine to the Tesla server. Since I am unable to use PuTTY, I use FileZilla and sign on to the Tesla server using “tesla.cs.iupui.edu” followed by my username and password. Upon signing in the tesla, I am able to transfer my files to the correct directory that I have labeled marketplace.



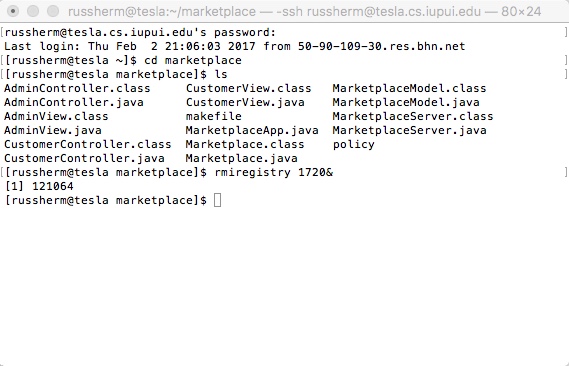
Now that my files are on the tesla server, I can open a terminal and connect to the tesla server via a remote connection.



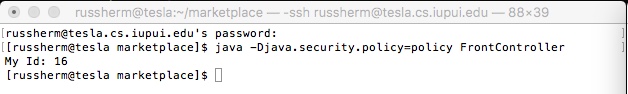
After logging in with my password, I am connected to the tesla server. I need to repeat these steps so I have two terminal windows open that are logged into tesla. I need to find the correct directory, and can do this by changing the directory to marketplace. Once I have done this, I can type ‘ls’ to see all the files in this directory. This will include all the files I just moved from my local machine to the tesla server.

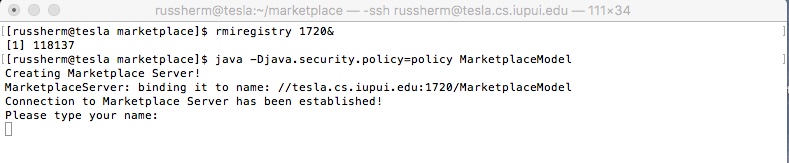


I need to make sure that the class files have all been compiled. I do this by typing “javac \*.java”. This will compile all of my files so they are ready to run. Next, I need to set up the Java RMI registry. I was able to hard code my registry port for port 1720. I can connect to this port by typing ‘rmiregistry 1720&’, with the & allowing the process to run in the background.

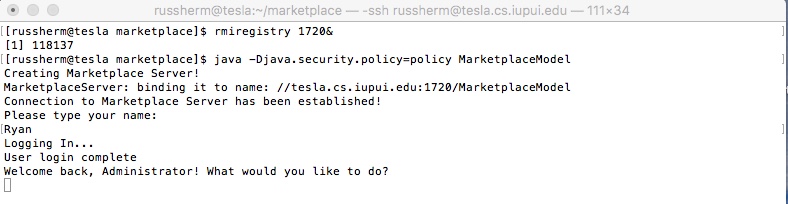


I know I am connected based on the “[1] 121064”. If the port had been in use, an Exception would have been thrown. Now that I have my port, I can run my server and client sides of my Java RMI connection. I will do this by calling my MarketplaceModel class, and also my newly created FrontController class. To run the MarketplaceModel file, I will enter the following “java -Djava.security.policy=policy MarketplaceModel”. I will do the same in the other terminal for the Customer Controller file, entering “java -Djava.security.policy=policy FrontController”.

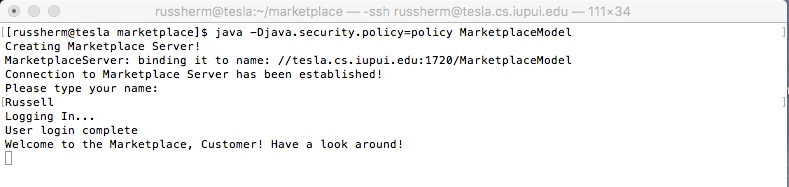




As we can see, the Marketplace Model establishes a Server and connects to it. It then asks for the user’s name. On the other side, the Front Controller connects to the JavaRMI and gives the user an Id number.



Entering the name “Ryan” will sign the user in as an Administrator and give the appropriate view.



Any other name will bring the user to the Customer view.