Assignment #3 Report

Russell Sherman

March 10, 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 #3

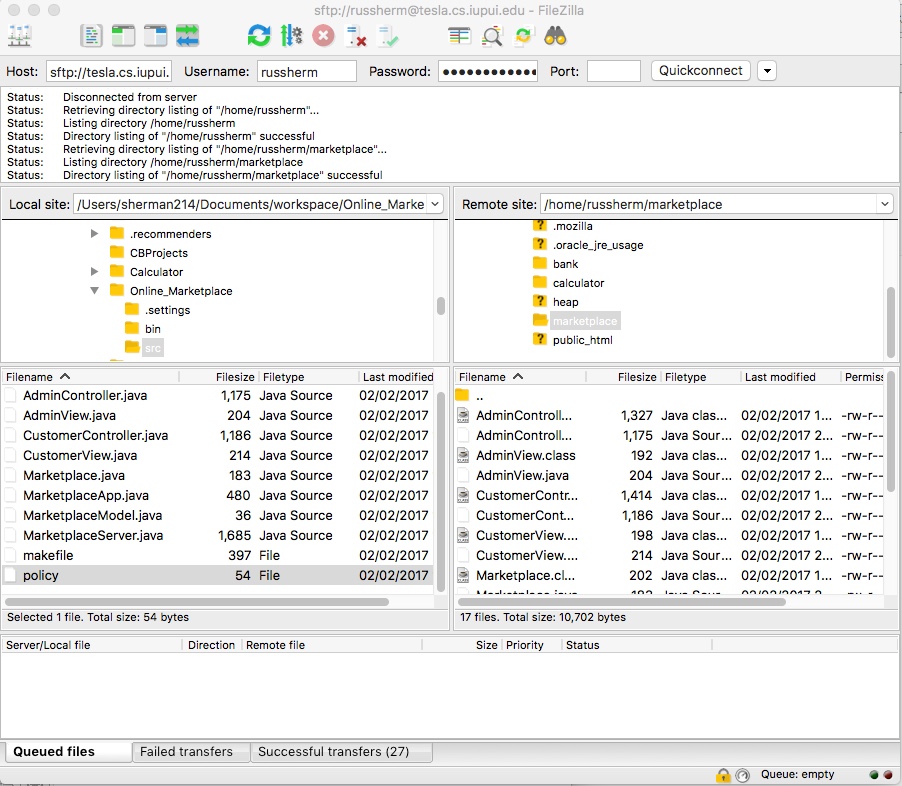
In assignment 3, we built upon our marketplace we developed for assignment 1 and 2. In this assignment, we used what we learned from class on the Authorization pattern, including role-based access control. We also looked to implement Java Annotations. In addition, we used Reflection and Proxy patterns.

Changes Made

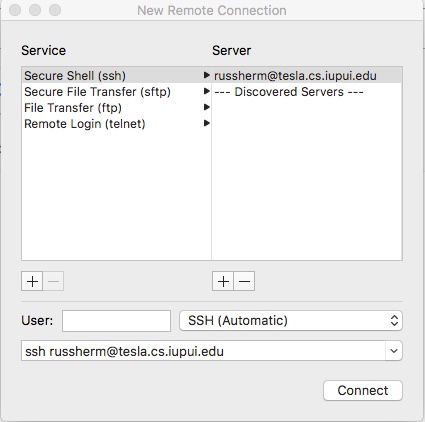
In this assignment, I revised my control pattern to add a number of new concrete commands, as well as an invoker, and command interface. This will allow me to implement my Java Annotations correctly. I also made some changes to my Java RMI. My application now starts with a Server and Client file. These files connect the two sides and then begin the application once a connection is established. I also created a RequiresRole class for deciding which role can access certain commands. The ServerImpl and ServerInterface also help to introduce Java Annotations.

Sample Run

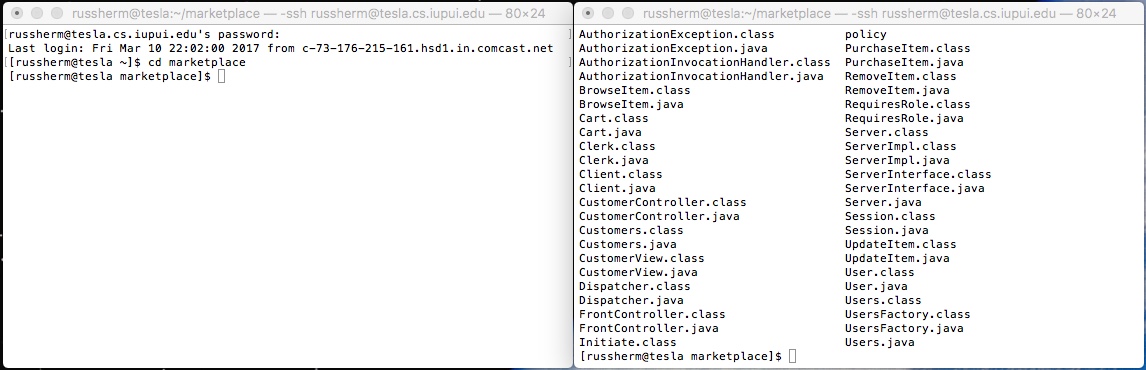
After saving my files in Eclipse, I open a program to transfer my files from my local machine to the Tesla server. Since I am unable to use PuTTY, I use FileZilla and sign on to the Tesla server using “tesla.cs.iupui.edu” followed by my username and password. Upon signing in the tesla, I am able to transfer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



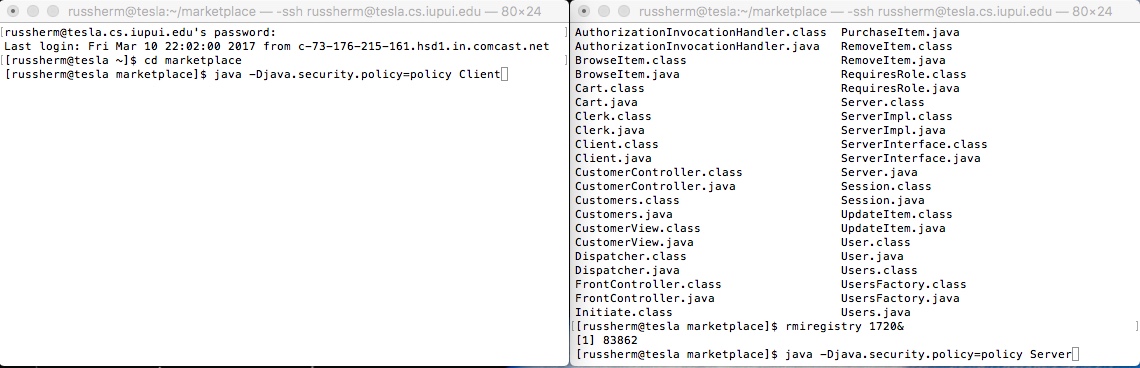
After logging in with my password, I am connected to the tesla server. I need to repeat these steps so I have two terminal windows open that are logged into tesla. I need to find the correct directory, and can do this by changing the directory to marketplace. Once I have done this, I can type ‘ls’ to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



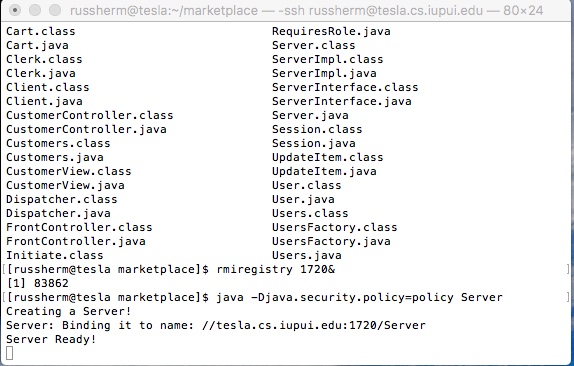
I need to make sure that the class files have all been compiled. I do this by typing “javac \*.java”. This will compile all of my files so they are ready to run. Next, I need to set up the Java RMI registry. I was able to hard code my registry port for port 1720. I can connect to this port by typing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



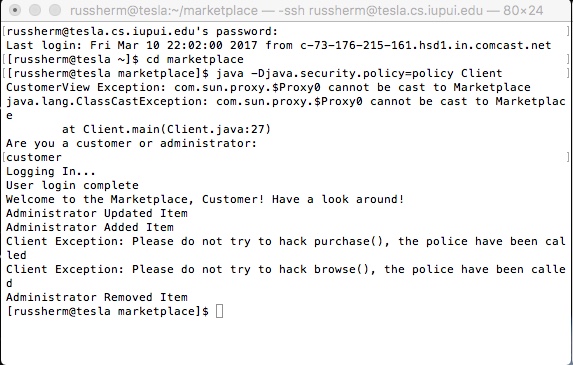
I know I am connected based on the “[1] 83862”. If the port had been in use, an Exception would have been thrown. Now that I have my port, I can run my server and client sides of my Java RMI connection. I will do this by calling my Server class, and also my newly created Client class. To run the Server file, I will enter the following “java -Djava.security.policy=policy Server”. I will do the same in the other terminal for the Customer Controller file, entering “java -Djava.security.policy=policy Client”.



As we can see, the Server establishes a connection and then the Client connects to the Java RMI. The Server will announce that it is creating a server, binding it to a name and then letting you know that the server is ready.



On the Client side, if run correctly, the user should be asked for their role in the marketplace. Upon choosing a role, they will be directed to a view, where it will display which commands they have access to. However, in my test run I received an Exception that I had an error with my Proxy. You can still see how that login is meant to work below.



You can sign in as customer or administrator currently, but eventually users will be assigned roles when they log in with usernames and passwords.