Monitor Stock Price

This mini project aims to build a dashboard shared by many users to keep a tab on the stocks they've purchased. And to achieve this objective, I've divided the project into manageable sub-modules:

1. Background service to keep polling the stock readings.
2. RESTful service to support the basic functionalities required.
3. UI to interact with the MySQL database and update the user changes.

Design and Approach

Database schema

1. Since it is a user based dashboard, I added an additional authentiaction (although sessions isn't yet in place), where the usernames are stored with the passwords (in plain text, not even hashed, let alone salting!) in the table .
2. And the background service that keeps polling the Yahoo Finances API, writes to the 'STOCKS' table that maintains the log of historic prices of all the stocks ever traded by any of our user.
3. To keep track of the user and his/her company we maintain a lightweight lookup that maps every users current inventory with the last updated stock price. So, UI rendering is responsive.

4. And like wise, since they are modeled to be different objects, the company data is abstaracted into a new table which maintains the latest values of the stock for a quick update.

Web Application

- I choose to build the web-application using SpringMVC framework primarily because of the vast online support and the fairly simple and consistently defined structure using views, models and controllers.
- And the implementations is straight forward, like a trypical MVC application, the data is modeled around the user and company objects. And to support the fucntionalites offered by the REST calls, services are defined and implemented.
- I’ve used several opensource libraries to make life easier, like plotly, yahoo-finance API (third-party).

Installation

- To start off with being able to run the package as is, we need to modify the configuration keys so that using the jdbc, it can talk to MySql.

  > *src/main/resources/config.properties*

StocksFetch

- Background service, that polls the data every 5 minutes.

> *mvn clean install; mvn exec:java*

UI

- Hosted using tomcat server.

  > *mvn clean install; mvn package; cp /target/*.war /usr/local/apache-tomcat/webapps*

Place the .war file generated in your apache-tomcat intallation directory. Fire up the tomcat server and you’ll be able to access it on http://localhost:8080/StockMonitoring/