COMP3322A MODERN TECHNOLOGIES ON WORLD WIDE WEB

Lab 5 jQuery and JSON

Overview

In this lab, we will implement the same web-based stock status system as what we implemented in Lab 4, using jQuery, plus achieving some additional functionality. Especially, we add a few input textboxes on the web page for adding a new stock entry.

Example screenshot of the page after initial loading is given in Fig. 1. The "Filter by Stockcode" and "Filter by Category" buttons work the same way as in Lab 4. Similar to Lab 4, when you click the status before each stock entry, it changes from RISE to FALL or vice versa.

In addition, on any page view, after filling in the input textboxes (e.g., Fig. 2) and clicking the "Add New Entry" button, the page will return to the full list display as in Fig. 1, including the newly added record (e.g., Fig. 3).

Setup Runtime Environment and MongoDB

Follow the instructions (Steps 1 to 3) in **setup_nodejs_runtime_and_examples_1.docx** to set up Node.js runtime environment and create an Express project named "**lab5**" (i.e., use "express lab5" in Step 3).

You can reuse the MongoDB database you have created for Lab 4, and launch the database server as follows:

- **Launch the 2nd terminal** (besides the one you use for running NPM commands), and switch to the directory where MongoDB is installed. Start MongoDB server using the "data" directory of "lab4" project as the database location, as follows: (replace "YourPath" by the actual path on your computer that leads to "lab4" directory)

```
./bin/mongod --dbpath YourPath/lab4/data
```

After starting the database server successfully, you should see some prompt in the terminal like "…2019-10-08T21:49:47.404+0800 I NETWORK [initandlisten] waiting for connections on port 27017..". This means that the database server is up running now and listening on the default port 27017. Then **leave this terminal open and do not close it when your Express app is running** in order to allow connections to the database from your Express app.

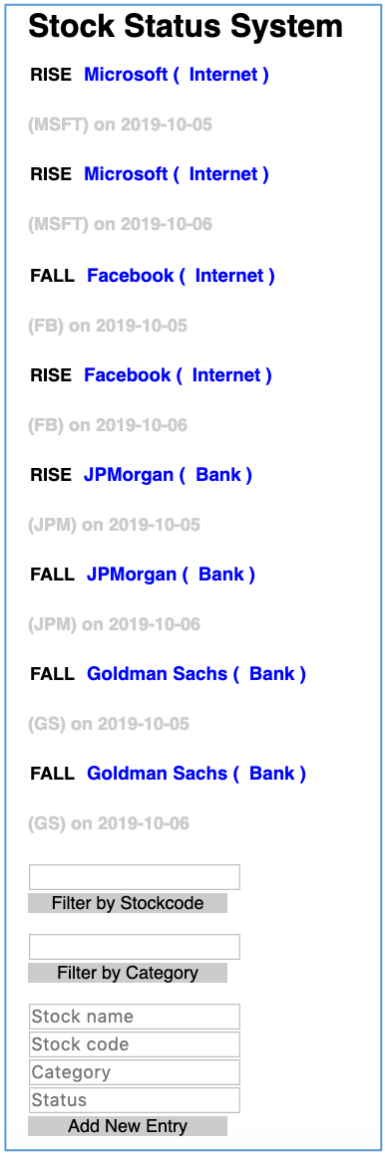

Fig. 1

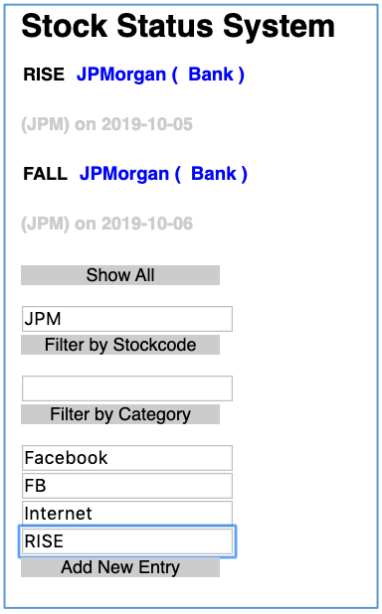

Fig. 2

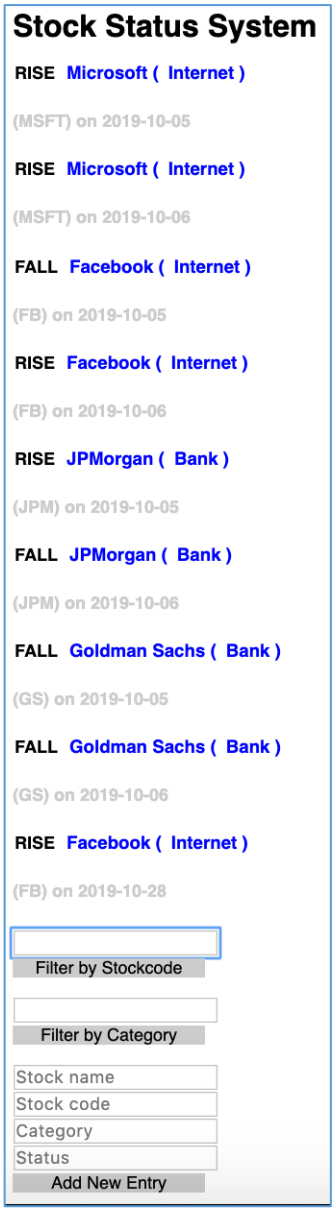

Fig. 3

- If you wish to manipulate documents in the "lab4" database, **launch the 3rd terminal**, switch to the directory where mongodb is installed, and execute the following commands:

```
./bin/mongo
use lab4
```

Then you can use db.stockList.find() to check out existing entries in the database, or use db.stockList.insert() to insert new documents as what you did in Lab 4.

Lab Exercise 1: re-implement Lab 4 functionalities using jQuery and JSON

Download **lab5_materials.zip** from Moodle. Unzip it and you will find three files needed for this app.

Copy **app.js** to the project folder "lab5" that you have created. Copy **index.html** to the "public" folder of the project directory and copy **style.css** to the "public/stylesheets" folder.

Open **app.js** with an editor and compare its code with app.js we implemented in lab4. You will find that in the middleware handling HTTP GET request for "/GetEntries", instead of sending constructed HTML content back to the client, here we directly send the retrieved documents (in JSON format) back to the client using res.json(docs). We add "console.log(JSON.stringify(docs))" in the code such that you can check out the content of the docs on the terminal where you run the app.

Open **index.html** with a text editor. Compare the code given in index.html and the code in index.html of Lab 4, you will find that we do not register event handlers using onload="xx" or onclick="xx" in the HTML part, but directly use jQuery APIs in <script>…</script> for event handling.

[Step1]: Implement the function **showAll()** using jQuery function $.getJSON (to issue an AJAX HTTP GET request for **GetEntries**) and $("#button_all").hide(); (for removing the "Show All" button from the page), to achieve the same functionality as the showAll() function in Lab 4.

We have provided a function showEntries() to parse the received JSON data and produce the HTML code for stock entry display, in the same format as the HTML content produced on the server side in Lab 4. You can call it to produce the HTML content for stock entry display in the element with id "entries".

[Step2]: Complete the callback function in $("#FS").click(function() { }); and $("#FC").click(function () { }); to achieve the same functionalities as in **filterS()** and **filterC()** in Lab 4. Use jQuery function $.getJSON to issue an AJAX HTTP GET request for **GetEntries** and other jQuery functions to obtain the input in the stockcode or category input textbox and to show the "Show All" button again.

[Step3]: Implement the function **changeState()** using jQuery methods to achieve the same functionality as changeState() in Lab 4. Especially, use jQuery AJAX function load() to issue an

AJAX HTTP POST request for **updateState** and display the received new status value in the current element.

Lab Exercise 2 – add new stock entry

Now we complete the client-side and server-side code for adding a new stock entry into the database.

[Step 1]: In **index.html**, implement the following callback function, which is invoked when the "Add New Entry" button is clicked. Send an AJAX HTTP POST request for "CreateEntry", enclosing the user entered stock name, category, code and status from the input textboxes. Upon receiving server response, clear the input textboxes for stock name, category, code and status (e.g., using $("#new_category").val("")), retrieve and display the complete list of stock entries including the newly added one (i.e., as in Fig. 3) by calling showAll().

```
$("#createEntry").click(function

});
```

[Step2]: In **app.js**, we provide an empty middleware for handling HTTP POST requests for "CreateEntries" as follow:

```
app.post('/CreateEntries', function(req, res){

})
```

Implement the middleware, which obtains stock name, status, code and category from the POST request, get the current time and extract the date (sample code provided below and refer to methods on JavaScript Date object here https://www.w3schools.com/js/js_date_methods.asp), and then use collection.insert() for inserting the new stock document into the database.

```
var d = new Date();
var date = d.getFullYear()+"-"+(d.getMonth()+1)+"-"+d.getDate();
```

Submission

You should submit the following files only:

(1) app.js
(2) index.html

Please compress the above two files in a .zip file and submit it on Moodle before 23:59 Wednesday Nov. 6, 2019:

(1) Login Moodle.
(2) Find “Labs” in the left column and click “Lab 5”.
(3) Click “Add submission”, browse your .zip file and save it. Done.
(4) You will receive an automatic confirmation email, if the submission was successful.
(5) You can “Edit submission” to your already submitted file, but ONLY before the deadline.