**Project ---- a web service for Adafruit products stock classification**

Gaomeizhu Qu

**Summary:**

The project here is a web service about a web crawler collecting product data from <https://www.adafruit.com/> and classifying them into three categories according to their sales condition, which is defined by the number of stock. The result of it displays on a web page and shown in three different tables. Each table includes products’ product id and the name, stock information and url.

**Description:**

This project uses Tomcat Apache as server in the back-end and collects raw data in JSON Array format from adafruit website via RESTful API provided by adafruit. Since the data from adafruit API is about 2M total, it will not bring the service overload without database. In order to maintain the real-time, this time it’s better to send processed data to front-end directly via API instead of getting data from database. The processed data is sent in JSON format and the value is three JSON array includes three kinds of product information. In the front-end, it goes a conciseness way, request data from server and displayed them into three tables.

**Back-end:**

**Server**: Tomcat v9.0, HTTP Servlet, while getting request from client, “doGET” call adafruit API and go through data processing and filter. And then write back to client via HTTP.

**Data process:** In order to avoid loading too much useless data information, before processing and passing through filter, I choose to reconstruct raw data to JSON format with the most useful information via an item builder pattern, it can also avoid null condition in some raw data tag.

**Logic:** Processed data iterated through the category filter and returned as Json array. The response sent from server is a Json object contains three Json arrays as result. Although the filter could be designed in the front end to reduce the load of back-end, it is better built in the back-end for the developer to collect user’s demand in the future and if it’s necessary, developer could improve the back-end and extract collected data for research. Also, while data is increasing, sending all data to the front-end to do the filter could have higher requirement for bandwidth. So it’s a trade-off condition and in this project, built in the back-end is better.

**Front-end:** While sending “GET” to server, the response returned in string format. The response is parsed in Json via JSON.parse() in order to get three categories, which turned into tables and shown on the website.

**Conclusion:**

Since the main task is expose the product information of best seller/well seller/common seller and finish the classification process, I kept front-end easy to show the result. Although this time, database is not necessary in order to keep the project’s real-time property and reduce the load of back-end. If the data increased in the further, it’s also a good idea to add database and keep the database refresh after a certain time (for example once an hour). For the further requirement, it’s easy to add more filters and more ajax interaction to this project.