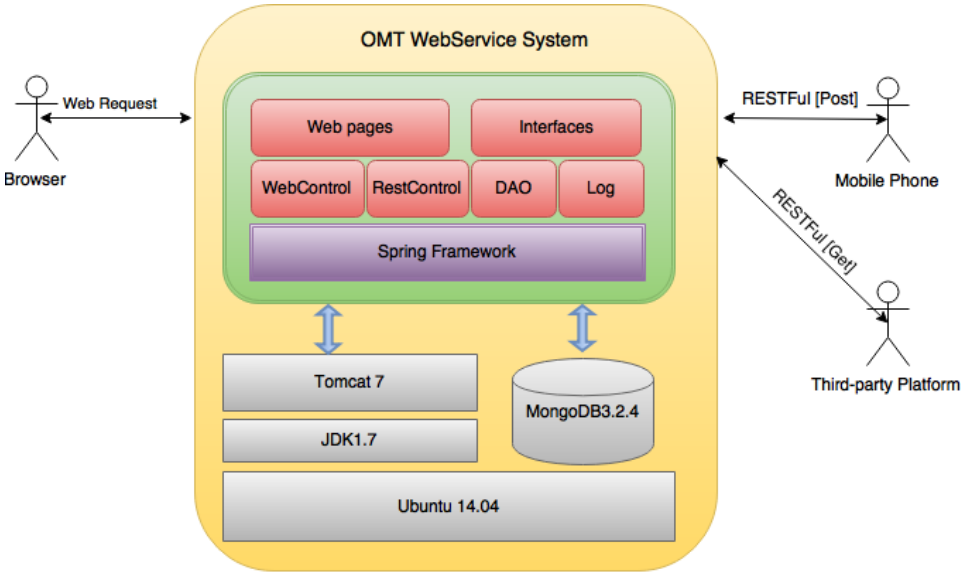
OMT-Web Service-Design

1. Summary

This document is written for explaining the OMT-Web Service Subsystem(OMTWSS), the structure, the environment, the tools and related developing process.

2. System Design

The OMTWSS was designed for both RESTful & Web Pages to realize the function of CRUD data from Webpage client and Mobile client.



- Platform: Linux (Ubuntu 14.04)
- Web Container: Tomcat 7 & JDK1.7
- Database: Mongodb 3.2.4 – NoSQL & Memory: Mass Data & More efficient
- Based on the popular framework: Spring
- Version & Compiler: Maven in Eclipse
- Dev IDE: Eclipse Mars
- Target Object: ROOT.war // to deduct the URL length

3. Web Service Interfaces:

Data Structure Usermessage:

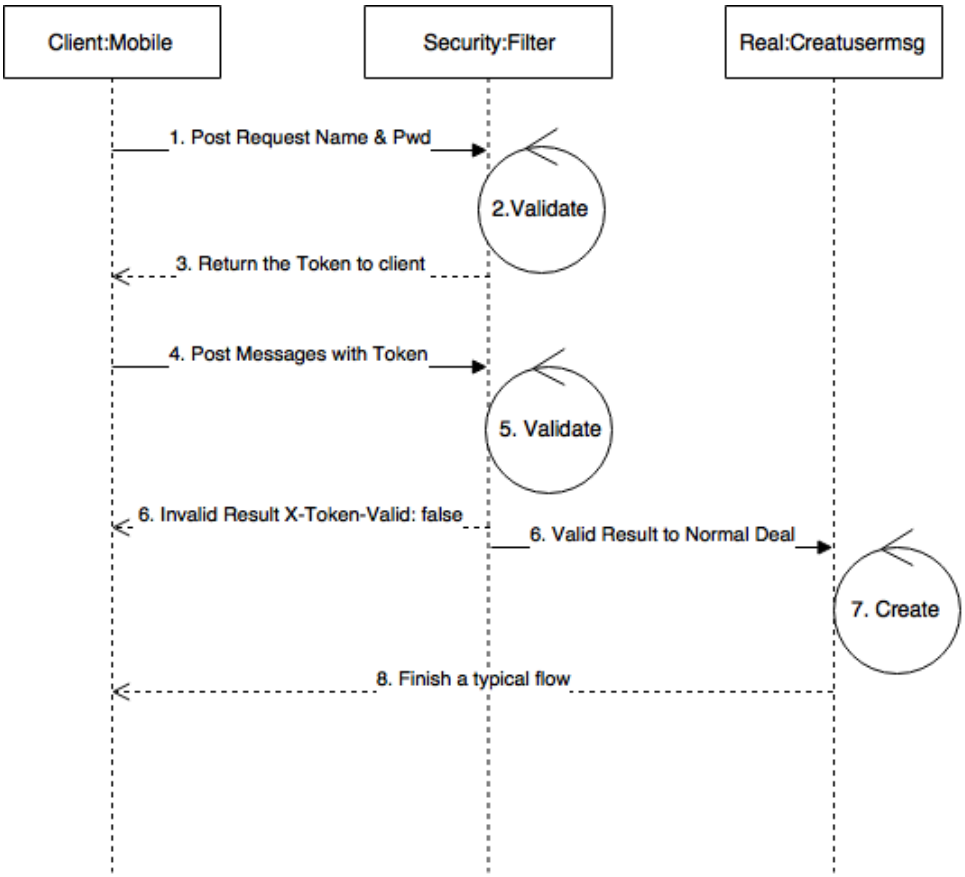
{"name":t.liu,"email":"t.liu@omnimarkettide.com","notes":"How are you?",datetime":"23/03/2016 10:12:00"}

The whole data record structure stored in mongodb:

```
public class UserMessage {
        private String id;  // records identify
```

```
    private String name;  // user name :must
    private String email; // user email :must
    private String notes; // user message :must
    private String datetime; // record time: :must
    private int status; // back end deal
}
```

Sequence Diagram:



Requests example details:

1. POST -H "X-Username: **{device_id}**" -H "X-Password: omt@mel123" http://query.omnimarkettide.com:15660/rest/ ; use this url to get a token from server side;
2. On each following post request together with the returned token: 8LwW+4p57oLgIjjmWdOjIzZFUjy+ku5LA4jB+KdKbzA=
3. Post: -H "X-Auth-Token: 8LwW+4p57oLgIjjmWdOjIzZFUjy+ku5LA4jB+KdKbzA=" http://query.omnimarkettide.com:15660/rest/secure/creatusermsg
4. Any token error will result in an error in HTTPResponse Header: X-Token-Valid: true|false

2. Web Socket Interface

- RESTful API

```
@RequestMapping(value="/agm/getuserhistmsg",
method=RequestMethod.POST, produces=MediaType.APPLICATION_JSON_VALUE)
```

----Get the history user messages with a email list

- Data Structure

  com.omt.webservice.entity.HistoryPureVO

  Com.omt.webservice.entity.MessageHistVO

- RESTful API wssreqhistory

```
@RequestMapping(value="/wssreqhistory", method=RequestMethod.POST,
produces=MediaType.APPLICATION_JSON_VALUE)
public @ResponseBody String wssreqhistory(@RequestBody HistRequestVO ruvo)
```

Data Structure:

Com.omt.websocket.entity.HistRequestVO

- RESTful API wssrequestcode

```
@RequestMapping(value="/wssocket/wssrequestcode",
method=RequestMethod.POST, produces=MediaType.APPLICATION_JSON_VALUE)
```

----Get the information with a code list

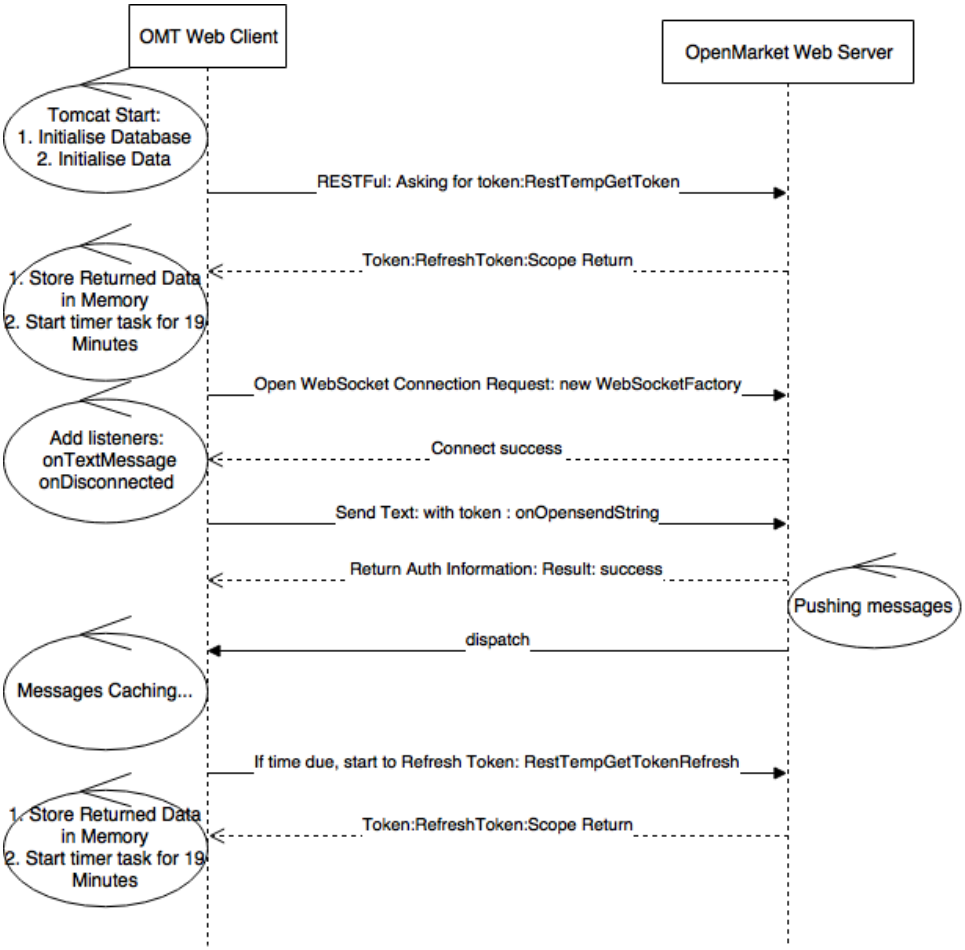
- Data Structure

  com.omt.websocket.entity.ShareData

  com.omt.websocket.entity.NotifyMessage

  Postman Example

- Process flow diagram



1. When tomcat server starts, after a while (10s), web socket client will do:
    i. Initialize database – load the instance operation object into memory
    ii. Initialize 5 items into table
2. Then this module will Send a RESTFul request with the following configuration to the websocket server:

```
public static final String URL_REST = "https://api.paritech.com/Paritech.AuthServer/OAuth2/Token";

public static final String PARITECH_USERNAME = "OMNILOOP";
public static final String PARITECH_PASSWORD = "qoz-why80";
public static final String PARITECH_GRANT_TYPE = "password";
public static final String PARITECH_GRANT_TYPE_REFRESH = "refresh_token";

public static final String PARITECH_CLIENT_ID = "omniloop";
public static final String PARITECH_CLIENT_SECRET = "nm27o9";

public static final String PARITECH_SCOPE = "http://api.paritech.com/wsapi/market/asset http://api.paritech.com/wsapi/market/depthfull http://api.paritech.com/wsapi

public static final String URI_WSS = "wss://wsapi2.paritech.com/Zenith"; //wss://wsapi2.paritech.com/Zenith [production URI]
public static final String PROTOCOL = "ZenithJson";
public static final String[] COMPANIES = {
        "{\"Controller\":\"Market\", \"Action\":\"Sub\", \"Topic\":\"Security!BHP.ASX\", \"TransactionID\":1, \"Data\":{\"Market\":\"ASX\", \"Code\":\"BHP\"}}",
        "{\"Controller\":\"Market\", \"Action\":\"Sub\", \"Topic\":\"Security!TLS.ASX\", \"TransactionID\":1, \"Data\":{\"Market\":\"ASX\", \"Code\":\"TLS\"}}",
        "{\"Controller\":\"Market\", \"Action\":\"Sub\", \"Topic\":\"Security!OMT.ASX\", \"TransactionID\":1, \"Data\":{\"Market\":\"ASX\", \"Code\":\"OMT\"}}",
        "{\"Controller\":\"Market\", \"Action\":\"Sub\", \"Topic\":\"Security!SUN.ASX\", \"TransactionID\":1, \"Data\":{\"Market\":\"ASX\", \"Code\":\"SUN\"}}",
        "{\"Controller\":\"Market\", \"Action\":\"Sub\", \"Topic\":\"Security!ONT.ASX\", \"TransactionID\":1, \"Data\":{\"Market\":\"ASX\", \"Code\":\"ONT\"}}"
};
```

3. Then this module will receive a JSON response for Token, Refresh_Token, and other things including expire time. This module will parse the response and store the token and refresh_token into memory with static varibles
4. Then a websocket object aimed to create a web socket connection will be created, together with adding: OnDisconnected and onTextMessage listeners
5. After the connection being successfully established, this module will send a onOpensendString to the websocket server which has been appointed for a valid connection.
6. If everything goes well, the websocket server will return the Connection Result[Success or Reject]
7. After the auth being successful, the websocket will constantly push messages to this module until the token on the server side expired. This module will tackle and parse all the messages and insert the needed data into mongodb

8. When the timer is due(19 minutes interval), this module will send the refresh_token to the web socket server to avoid the connection being expired, and start a new timer task at the same time.
9.

4. Map the IP to domain name

query.omnimarkettide.com -- 52.63.146.118 [Physical layer], then:

a. Sudo vi /etc/hosts to add a line query.omnimarkettide.com 52.63.146.118
b. Sudo vi /var/lib/tomcat7/conf/server.xml modify host to: query.omnimarkettide.com and port number to 15660 as well
c. Change the package name to ROOT.war
d. Sudo service tomcat7 stop; sudo service tomcat7 start
e. Working url: http://query.omnimarkettide.com:15660/agm/index

5. Database Backup

a. Backup whole database:
   i. Mkdir mongobak
   ii. Cd mongobak
   iii. mongodump

   Bakpath: ./dump/[databasename]/[collectionname].bson

a. Backup the specified database
   i. mongodump -d omtdb
   ii. Bakpath: ./dump/[databasename]/[collectionname].bson

b. Database restore
   i. Cd mongobak
   ii. Mongorestore --drop :// restore all data and drop the data in db, otherwise there would be some repeat records.
   iii. mongorestore -d omtdb --drop //just restore the omtdb
   iv. Mongorestore -d omtdb –c userMessage –drop ;// just restore collection

c. Import data to Mongodb
   i. mongoimport -d omtdb -c page --type csv --headerline --drop < csvORtsvFile.csv ;// type include: csv,json,tsv
   ii. mongoimport -d omtdb -c historyPriceVO -j 4 --type json --drop <historyPriceVO.json
   iii.

d. Export data from Mongodb
   i. mongoexport -d omtdb -c userMessage -q {} -f _id,name,email,datetime,notes --type=csv > pages.csv
   ii. mongoexport -d omtdb -c historyPriceVO -q {} --type=json >historyPriceVO.json

Mongodb batch update:

db.getCollection('historyPriceVO').update({ date: { $eq: "06/05/2016" } },{ $set: { date: "05/05/2016" } },{ multi: true })

mongoexport -d omtdb -c stPageShowVO -q {$group:{_id:"$code"}} --type=json >companies.json

http://www.mkyong.com/mongodb/mongodb-aggregate-and-group-example/

mongo command

> var **groupcode**=db.stPageShowVO.aggregate({$group:{_id:"$code"}})

> db.companyList.insert(**groupcode**.toArray())

6. References

a. Install JDK & tomcat 7

- sudo apt-get update
- sudo apt-get install tomcat7
- When start tomcat fail use these command:
- **sudo dpkg-reconfigure tomcat7**

b. Install Mongodb3.2.4

- sudo apt-key adv --keyserver hkp://keyserver.ubuntu.com:80 --recv 7F0CEB10
- echo "deb http://repo.mongodb.org/apt/ubuntu "$(lsb_release -sc)"/mongodb-org/3.0 multiverse" |
- sudo tee /etc/apt/sources.list.d/mongodb-org-3.0.list
- sudo apt-get update
- sudo apt-get install -y mongodb-org

c. Connect to AWS instance:

ssh -i "WebService01Key.pem" ubuntu@ec2-52-63-146-118.ap-southeast-2.compute.amazonaws.com

d. Transfer files to AWS instance:

scp -i WebService01Key.pem omtwebservice.war ubuntu@ec2-52-63-146-118.ap-southeast-2.compute.amazonaws.com:~

a. scp -i WebService01Key.pem ubuntu@ec2-52-63-146-118.ap-southeast-2.compute.amazonaws.com:/var/lib/tomcat7/logs/catalina.out.gz ~/Downloads/
b. Use draw.io to draw online and export the images.
c. Check tomcat logs with Grep:

```
tail -f catalina.out  | grep -vE '(ReadingThread|received)'
```

tail -f omniinfo.log | grep currentHisPriceHt

tail -f omniinfo.log | grep -i 'currentHisPriceHt'

tail -f omniinfo.log | grep -iE '(currentHisPriceHt|code)'

1. Tcpdump command:
2. The following command with option -XX capture the data of each packet, including its link level header in HEXand ASCII format.

tcpdump -XX -i eth0 src 52.63.30.26

As we said, that tcpdump has a feature to capture and save the file in a .pcap format, to do this just execute command with -w option

tcpdump -w 0001.pcap -i eth0 src 52.63.30.26

To read and analyze captured packet 0001.pcap file use the command with -r option, as shown below.

tcpdump -r 0001.pcap

Top –o MEM [check memory usage sorted by mem]

To check MongoDB status:

db.serverStatus()

Check the size of a folder:

ubuntu@ip-10-0-3-144:/$ sudo du -hs /var/lib/

2.0G /var/lib/

ubuntu@ip-10-0-3-144:/$ df -Th

Filesystem Type Size Used Avail Use% Mounted on

udev devtmpfs 492M 12K 492M 1% /dev

tmpfs tmpfs 100M 348K 99M 1% /run

/dev/xvda1 ext4 7.8G 5.6G 1.8G 77% /

none tmpfs 4.0K 0 4.0K 0% /sys/fs/cgroup

none tmpfs 5.0M 0 5.0M 0% /run/lock

none tmpfs 497M 0 497M 0% /run/shm

none tmpfs 100M 0 100M 0% /run/user

ubuntu@ip-10-0-3-144:/$ lsblk

NAME MAJ:MIN RM SIZE RO TYPE MOUNTPOINT

xvda 202:0 0 8G 0 disk

└─xvda1 202:1 0 8G 0 part /

ubuntu@ip-10-0-3-144:/$

1. Linux Command lines:
    a. lsof -i:8887 [check the information regarding port 8887]

    mongoimport -d omtdb -c historyPriceVO -j 8 --type json --drop <historyPriceVO.json

mongoexport -d omtdb -c companyList -q {} --type=json > companyList.json

mongoimport -d omtdb -c companyList -j 4 --type json --drop <companyList.json

scp -i WebService01Key.pem ubuntu@ec2-52-63-146-118.ap-southeast-2.compute.amazonaws.com:/home/ubuntu/companyList.json ~/Downloads/

http://msxml.tenfore.com/index.php?username=user&password=pass&instrument=151.1.VOD

http://msxml.tenfore.com/index.php?username=username&password=password&JSONShort&instrument=146.1.TLS

mongoexport -d omtdb -c -q {} --type=json > .json

scp -i WebService01Key.pem ubuntu@ec2-52-63-146-118.ap-southeast-2.compute.amazonaws.com:/home/ubuntu/ ~/Downloads/

mongoimport -d omtdb -c -j 4 --type json --drop < .json

scp -i WebService01Key.pem ROOT.war ubuntu@ec2-52-63-146-118.ap-southeast-2.compute.amazonaws.com:~

> db.userMessage.count()

86

> **db.userMessage.aggregate([{"$group":{_id:{name:"$name",status:"$status"},count:{$sum:1}}}])**

{ "_id" : { "name" : "", "status" : 0 }, "count" : 1 }

{ "_id" : { "name" : "Reid Skip", "status" : 0 }, "count" : 27 }

{ "_id" : { "name" : "", "status" : 3 }, "count" : 1 }

{ "_id" : { "name" : "Richard Dennis", "status" : 1 }, "count" : 4 }

{ "_id" : { "status" : 1 }, "count" : 4 }

{ "_id" : { "name" : "Richard Dennis", "status" : 0 }, "count" : 1 }

{ "_id" : { "name" : "Ross Blair-Holt", "status" : 1 }, "count" : 4 }

{ "_id" : { "name" : "oliver kidd", "status" : 1 }, "count" : 1 }

{ "_id" : { "name" : "Andrew Keys", "status" : 1 }, "count" : 7 }

{ "_id" : { "name" : "Megan Walker", "status" : 1 }, "count" : 1 }

{ "_id" : { "name" : "", "status" : 1 }, "count" : 15 }

{ "_id" : { "name" : "Scott Standen", "status" : 1 }, "count" : 3 }

{ "_id" : { "name" : "OMT Test", "status" : 1 }, "count" : 1 }

{ "_id" : { "name" : "Reid Skip", "status" : 1 }, "count" : 12 }

{ "_id" : { "name" : "Kapil Shah", "status" : 1 }, "count" : 1 }

{ "_id" : { "name" : "Phil Avery", "status" : 1 }, "count" : 2 }

{ "_id" : { "name" : "megan boston", "status" : 1 }, "count" : 1 }

db.userMessage.aggregate([{"$group":{_id:{name:"$name"},count:{$sum:1}}},{$sort:{"count":-1}}])

{ "_id" : { "name" : "Reid Skip" }, "count" : 39 }

{ "_id" : { "name" : "" }, "count" : 17 }

{ "_id" : { "name" : "Andrew Keys" }, "count" : 7 }

{ "_id" : { "name" : "Richard Dennis" }, "count" : 5 }

{ "_id" : { "name" : null }, "count" : 4 }

{ "_id" : { "name" : "Ross Blair-Holt" }, "count" : 4 }

{ "_id" : { "name" : "Scott Standen" }, "count" : 3 }

{ "_id" : { "name" : "Phil Avery" }, "count" : 2 }

{ "_id" : { "name" : "Kapil Shah" }, "count" : 1 }

{ "_id" : { "name" : "OMT Test" }, "count" : 1 }

{ "_id" : { "name" : "Megan Walker" }, "count" : 1 }

{ "_id" : { "name" : "megan boston" }, "count" : 1 }

{ "_id" : { "name" : "oliver kidd" }, "count" : 1 }