Subscribe Socket Request For MarketIDs

Document History

No.	Date of Revision	Version No.	Description of Change	Author
1.	03/12/2021	1.0	Initial Draft	Gurubani Gulati

Table of Contents

1. Introduction

To subscribe to the sockets to create the respective JSON files for the given markets id's.

1.1 Analysis:

- Analyze the folder provides as **CenterRateSignalr_Display**
- Run the **index.html**
- On the browser, a screen will be prompted wherein when you enter the respective market id and respective to this the market data is received.

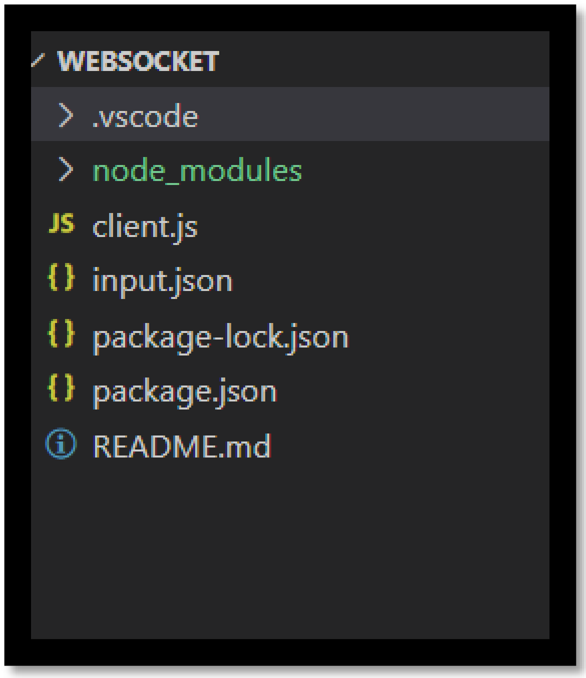
1.2 Objective:

The goal specializes in the below-listed points:

- Create the **NodeJs** script.
- Fetch Market id with **pmid** variable from input.json
- Code to subscribe the socket using **signalr**
- Iterate the **pmid's** over the connection
- After getting response from apiStructure the output json object
- Store the response in respective **<marketId>.json**

2. Development:

This is the folder with the name: **WebSocket.** The below is the folder structure for the NodeJS script.



```
WEBSOCKET
  > .vscode
  > node_modules
  client.js
  input.json
  package-lock.json
  package.json
  README.md
```

Client.js: Contains the actual code for the script.

Input.json: This is the JSON file that contains the pmids. This is the input file that is consumed by the scripts (client.js) to get all the respective market Ids.

Package-lock.json: These JSON files get created on the npm install command

Package.json: This JSON file has all the packages which are installed to run the script.

README.md: Contains the information to start and run the project on the respective peripheral.

2.1 Client.js

Install the respective packages as in express, fs, colors, @microsoft/signalr

- **express:** To use the client in a NodeJS application, install the package to your node_modules folder and use require(‘express) to load the module.
- **fs:** To use the client in a NodeJS application, install the package to your node_modules folder and use require(‘fs’) to load the module. This is used for file operations read/ write.
- **colors:** To use the client in a NodeJS application, install the package to your node_modules folder and use require(‘colors’) to load the module. This is used to color your console logs.
- **@microsoft/signalr:** To use the client in a NodeJS application, install the package to your node_modules folder and use require('@microsoft/signalr') to load the module. The object returned by require('@microsoft/signalr') has the same members as the global signalR object (when used in a browser).

```
var express = require('express');
var app = express();
var fs = require("fs");
const colors = require('colors');
const signalR = require("@microsoft/signalr");
```

vUri: the vUri is a variable that contains the URL for the endpoint over which the connection will be formed for sending and receiving data. Express is just a module framework for Node that you can use for applications that are based on server/s that will "listen" for any input/connection requests from clients.

```
var vUri = "http://a112.thebetmarket.com/SignalR";
```

The GET function call of the end point: http://localhost:8081/ , will hit the below function. For subscribing to socket we have used the @microsoft/signalr package. This will create a connection using signalR.HubConnectionBuilder function wherein:

- **.configureLogging**: will log all the events during the process
- **.withUrl** will be responsible for building and communicating with the url(vUri) provided to send and receive data
- **withAUtomaticReconnect:** will be responsible to automatic reconnect if and when the connection breaks at any point of time.
- **build:** will build the connection.

Post this connection.start will be called. This will read the input.json file and get the respective market ids to iterate (pmid). As a next step will invoke the connection on Rate and communicate with url and receive the data. The response of the api call will write the data. Furthermore, this data will be revise and make it in the presentable format and finally store in respective **<marketId>.json**

```
app.get('/', function (request, response) {

    var connection = new signalR.HubConnectionBuilder()
        .configureLogging(signalR.LogLevel.Trace)
        .withUrl(vUri)
        .withAutomaticReconnect()
        .build();


    connection.on('Rate', function (marketrate) {

        const data = JSON.stringify(marketrate, null, 4);
        response.write(data);

        var marketData = {};
        var events = [];

        marketData['mi'] = marketrate['mi'];
        marketData['ms'] = marketrate['ms'];
        marketData['tm'] = marketrate['tm'];


        for (var key in marketrate['rt']) {

            var eventsResponse = {};
            var backCount = 0;
            var layCount = 0;
```

```
 if (!(events.find(mid => mid.SelectionId == marketrate['rt'][key]['si']))) {

               eventsResponse['SelectionId'] = marketrate['rt'][key]['si'];
               eventsResponse['BackPrice1'] = 0.0
               eventsResponse['BackSize1'] = 0.0
               eventsResponse['BackPrice2'] = 0.0
               eventsResponse['BackSize2'] = 0.0
               eventsResponse['BackPrice3'] = 0.0
               eventsResponse['BackSize3'] = 0.0
               eventsResponse['LayPrice1'] = 0.0
               eventsResponse['LaySize1'] = 0.0
               eventsResponse['LayPrice2'] = 0.0
               eventsResponse['LaySize2'] = 0.0
               eventsResponse['LayPrice3'] = 0.0
               eventsResponse['LaySize3'] = 0.0

               for (var i in marketrate['rt']) {

                  if (eventsResponse['SelectionId'] ==
marketrate['rt'][i]['si']) {

                       if (marketrate['rt'][i]['ib'] == true) {
                           backCount++;
                           eventsResponse['BackPrice' + backCount] =
marketrate['rt'][i]['re'];
                           eventsResponse['BackSize' + backCount] =
marketrate['rt'][i]['rv'];
                       }
                       else {
                           layCount++;
                           eventsResponse['LayPrice' + layCount] =
marketrate['rt'][i]['re'];
                           eventsResponse['LaySize' + layCount] =
marketrate['rt'][i]['rv'];
                       }

                   }

               }

               events.push(eventsResponse);
           }
```

```
        }


        marketData['events'] = events;


        try {

            fs.writeFileSync(marketrate['mi'] + '.json',
JSON.stringify(marketData, null, 4));
            console.log(`${marketrate["mi"]} JSON data is saved.`.green);

        } catch (error) {
            console.error(error);
        }

    });


    connection.start()
        .then(function (vobj) {

            fs.readFile('input.json', (err, data) => {

                if (err) throw err;

                let marketIds = JSON.parse(data);

                for (var id in marketIds) {
                    connection.invoke('ConnectMarketRate',marketIds[id]['pmid']);
                }

            });
        })
        .catch(error => {
            console.error(error.message);
        });


});
```

Start Server: To start the server, the below code is responsible. This will listen the port 8081. This port can be changed as per convenience.

```
var server = app.listen(8081, function () {
    var host = server.address().address
    var port = server.address().port
    console.log("Example app listening at http://%s:%s", host, port)
})
```

3. Run:

To run the respective standalone independent scripts. Project code can be downloaded or cloned from GitHub repo: https://github.com/gurbanigulati17/websocket-signalr-nodejs

- Open the terminal.
- Type: npm install, and hit ENTER.
- Type: **node client.js**, and hit ENTER.
- Open the browser, and visit: http://localhost:8081/
- As a result, respective **<marketId>.json** will be created in the folder location of the project.

4. FAQ:

If there is any faq, please let us know.