Log Analyzer

v1.0

Readme document

Author: Kartik Kamath

Contents:

1. Softwares used
2. Pre-requisites
3. Using the application
4. Approach
5. Assumptions
6. Possible improvements

Softwares used:

1. Server side: Spring Boot, Apache Maven, log4j, ActiveMQ, Java8
2. Client side: jQuery, Bootstrap, DataTables, CSS, JavaScript, Plupload
3. IDE: Eclipse (Version: Oxygen.3a Release(4.7.3a)) (or Spring Tool Suite)
4. Database: MySQL

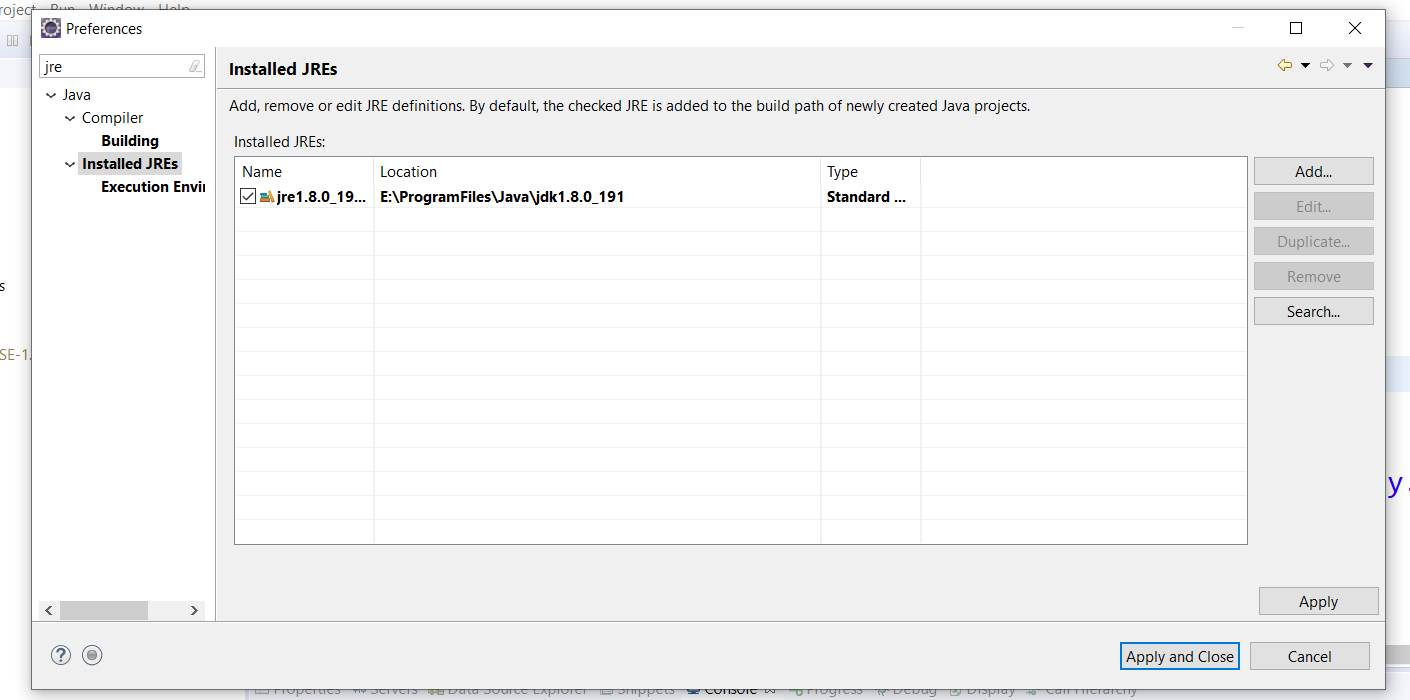
Pre-requisites:

1. Java8 must be installed

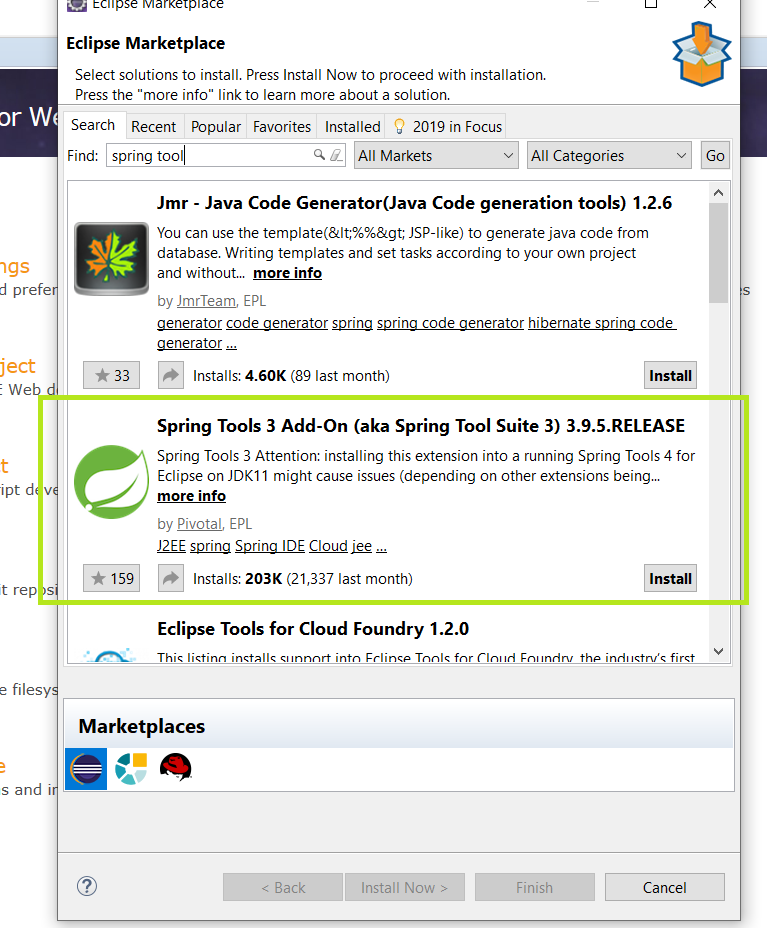
After installation, set environment variable JAVA\_HOME to your jdk path

1. Eclipse Oxygen configuration

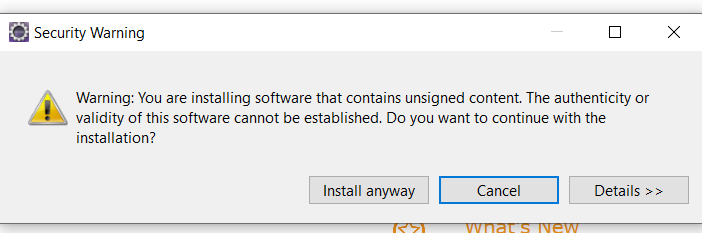
* Setup java path in eclipse to the jdk of the system (Windows -> Preferences -> Java-> Installed jres section)



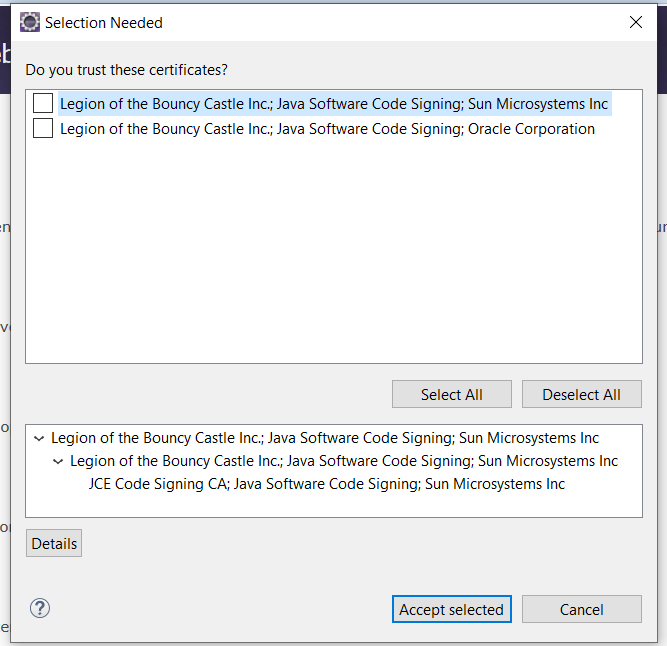
* Install Sts plugin from eclipse marketplace



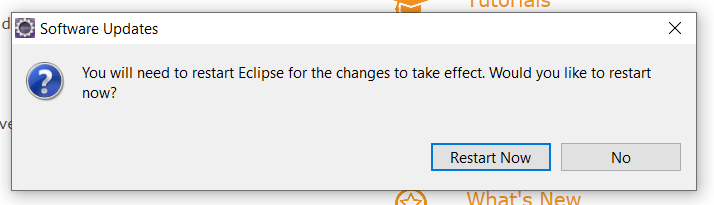
* Confirm, accept all terms and conditions.
* Security warnings can be ignored. Click Install anyway



* Select all certificates and accept

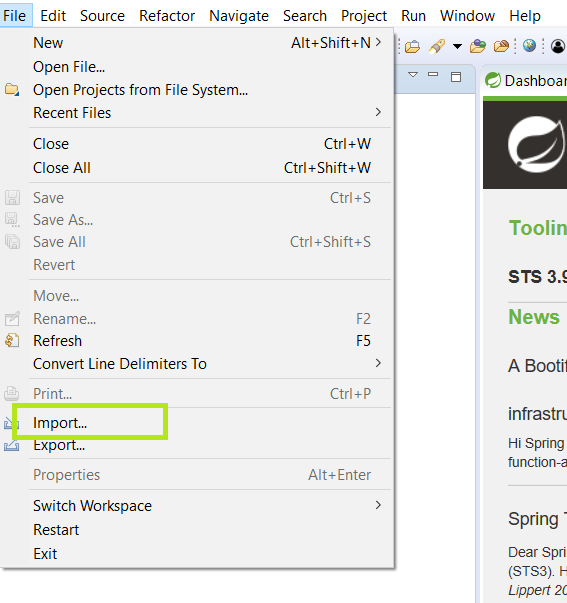


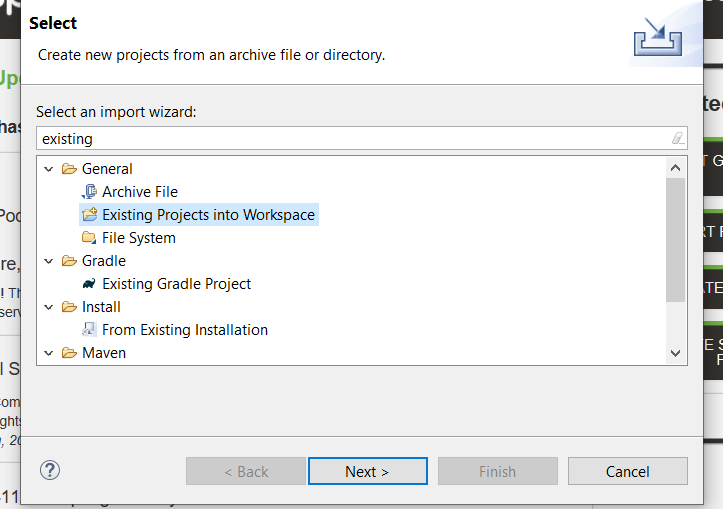
* Restart eclipse, if such message appears



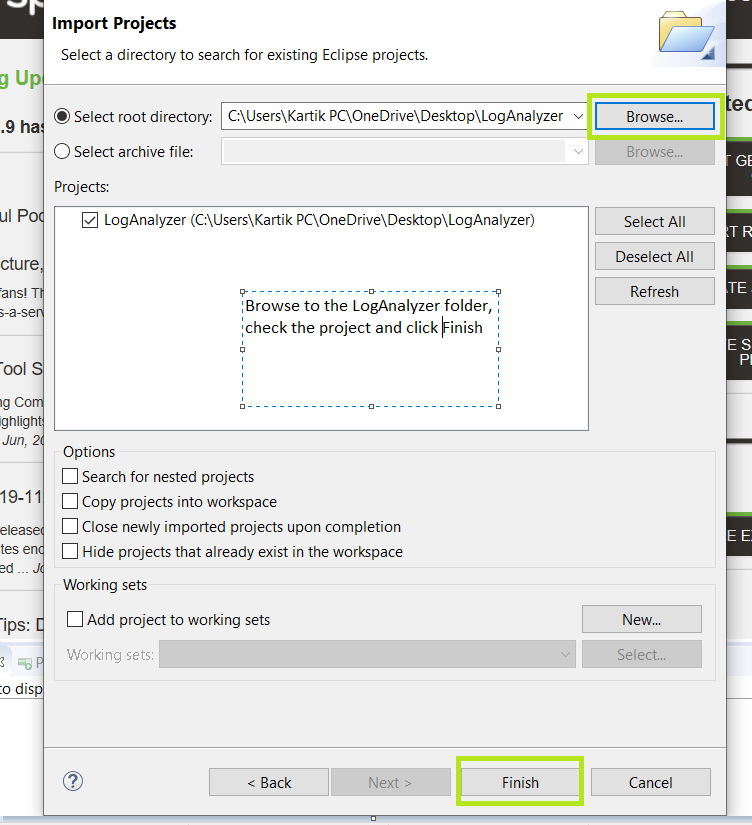
* If plugin is not available, use Spring Tool Suite IDE and start from first step and exclude this plugin installation step
* Setup maven
* To install Maven on windows, use apache-maven-3.6.1-bin.zip provided in the solution zip
* Unzip it to the folder you want Maven to live
* Add both *M2\_HOME* and *MAVEN\_HOME* variables to the Windows environment using system properties, and point it to your Maven folder
* Update the PATH variable by appending the Maven bin folder – *%M2\_HOME%\bin*, so that you can run the Maven’s command everywhere
* To verify it run: mvn -version in the command prompt. It should display the Maven version, the java version, and the operating system information

1. Importing project from folder zip (steps in screenshots)





Select the LogAnalyzer folder while browsing



1. Starting ActiveMQ

* Extract apache-activemq-5.14.2-bin.zip to a folder of your choice
* After extracting, go to this path

apache-activemq-5.14.2/bin/win64 (for 64-bit systems) OR

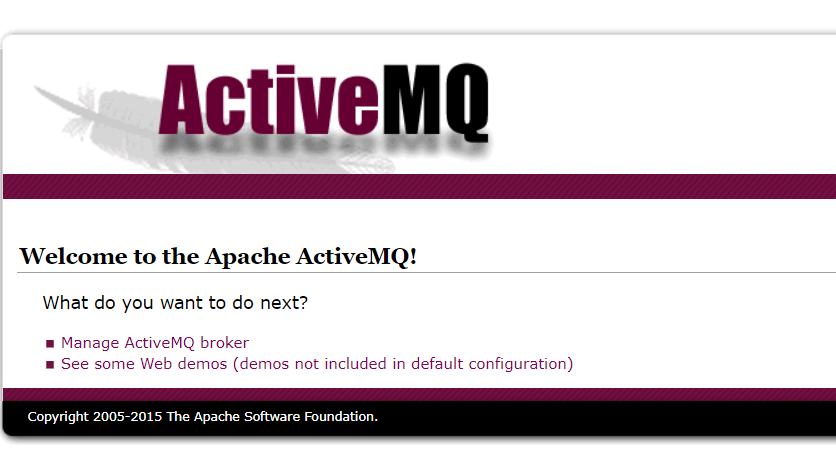
apache-activemq-5.14.2/bin/win32 (for 32-bit systems)

* Click activemq.bat. A command console opens up to start ActiveMQ
* After a few minutes, visit <http://localhost:8161>

If browser asks for credentials:

username: admin

password: admin



If such screen is shown, ActiveMQ has started successfully

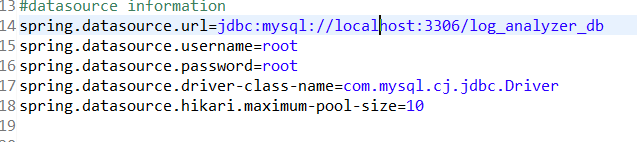
1. Setting up the database for MySQL

* Install MySQL from a vendor of your choice
* Execute log\_analyzer\_db\_queries.sql in your mysql to create the database and necessary tables
* Some sample data would have been inserted in the table, for execution flow purpose
* The mysql username and password currently used is:

Username : root

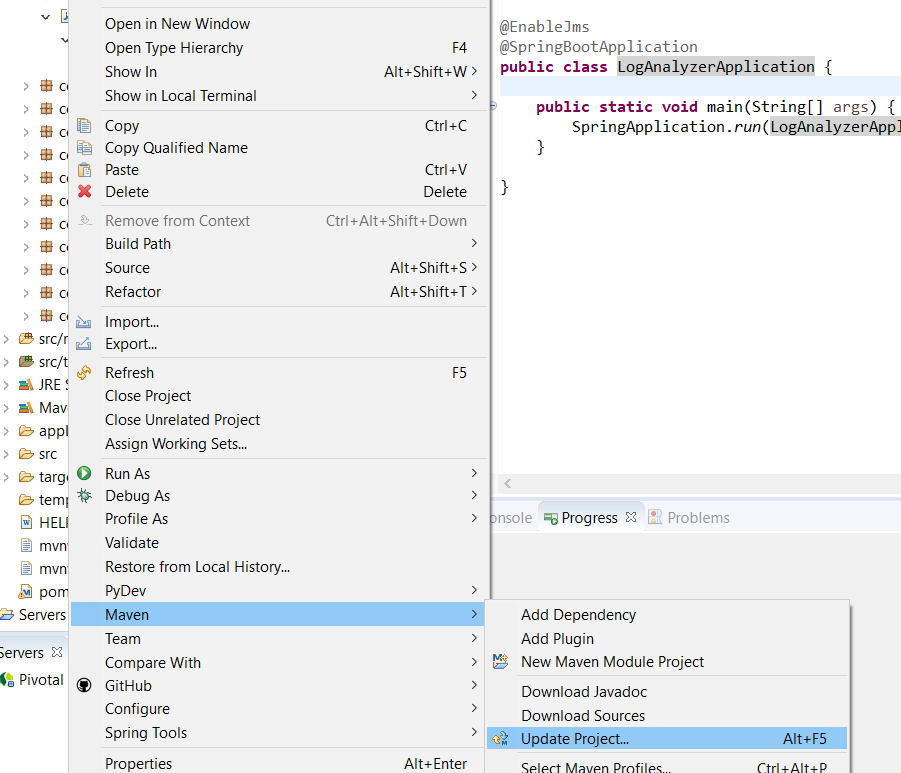
Password: root

If some other credentials are being used, changes have to be made in src/main/resources/application.properties :

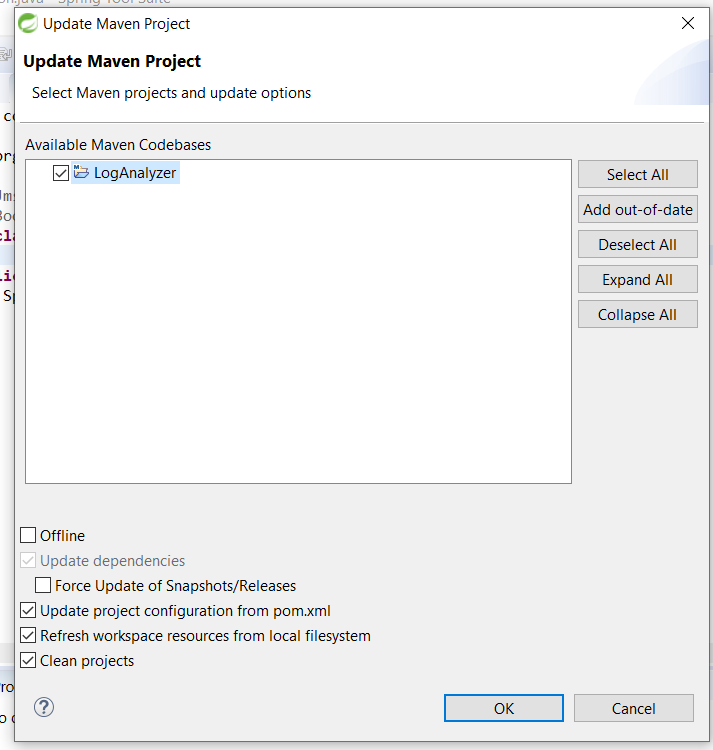


1. Booting the project

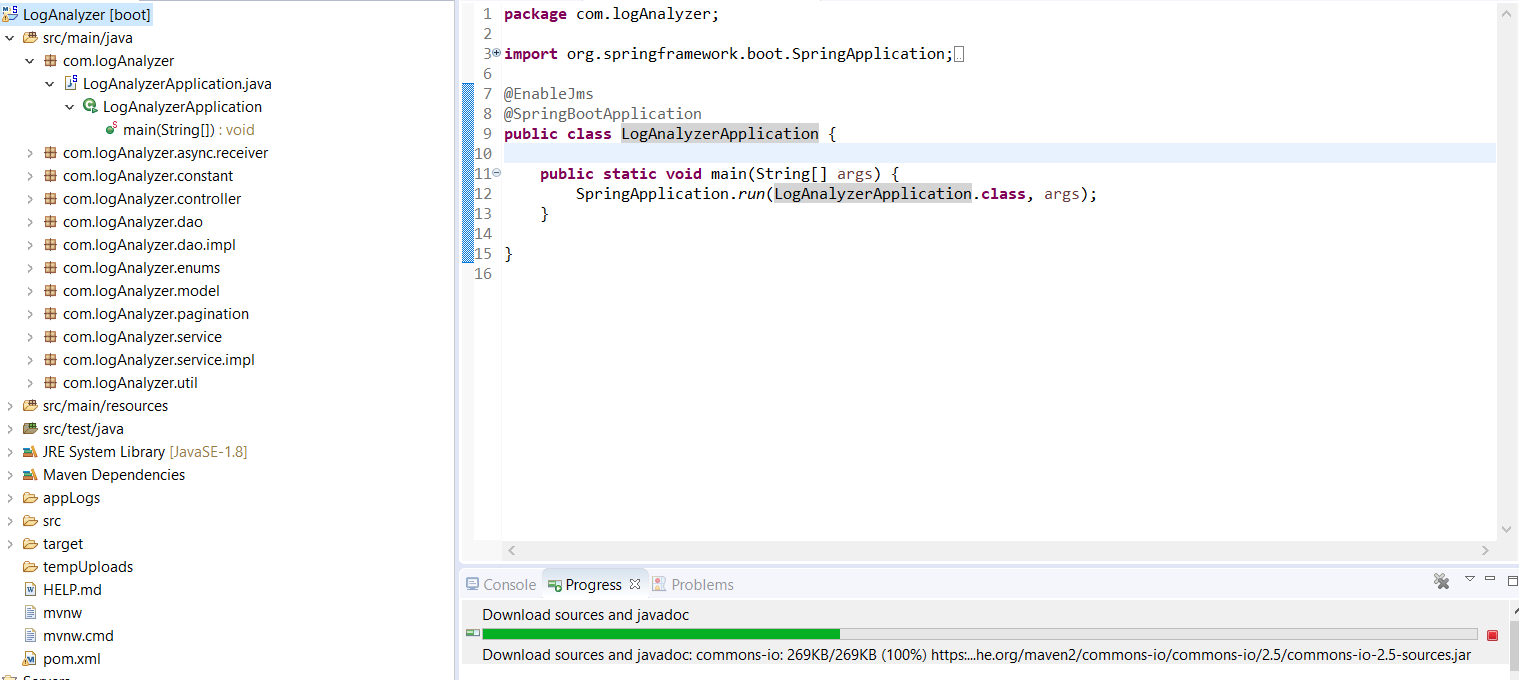
* The project needs to have all the dependencies before starting up. For this, we need to update the maven project by right clicking on project name -> Maven -> Update Project



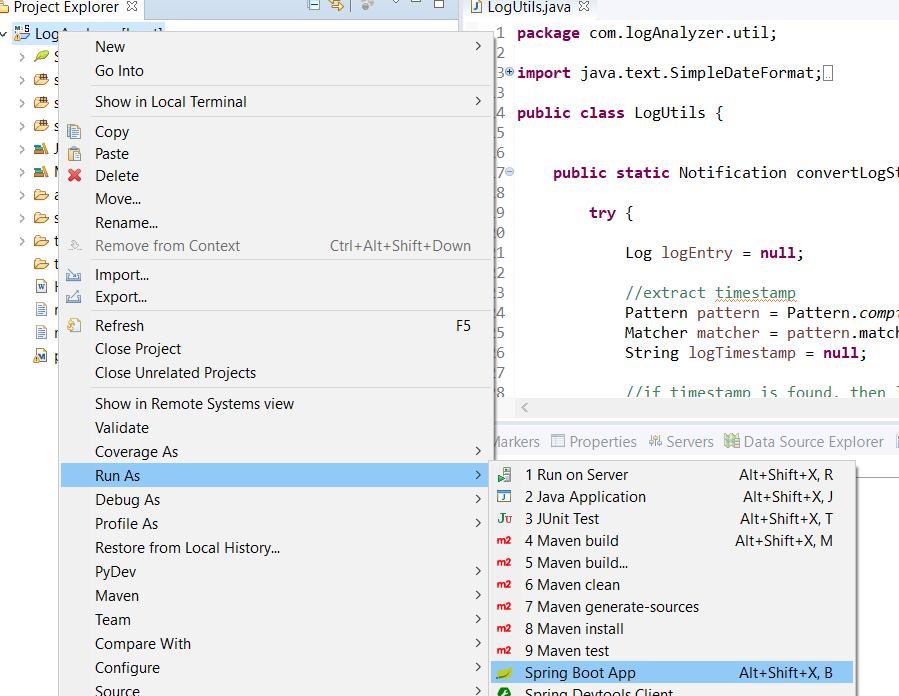
* Select the project name and click OK



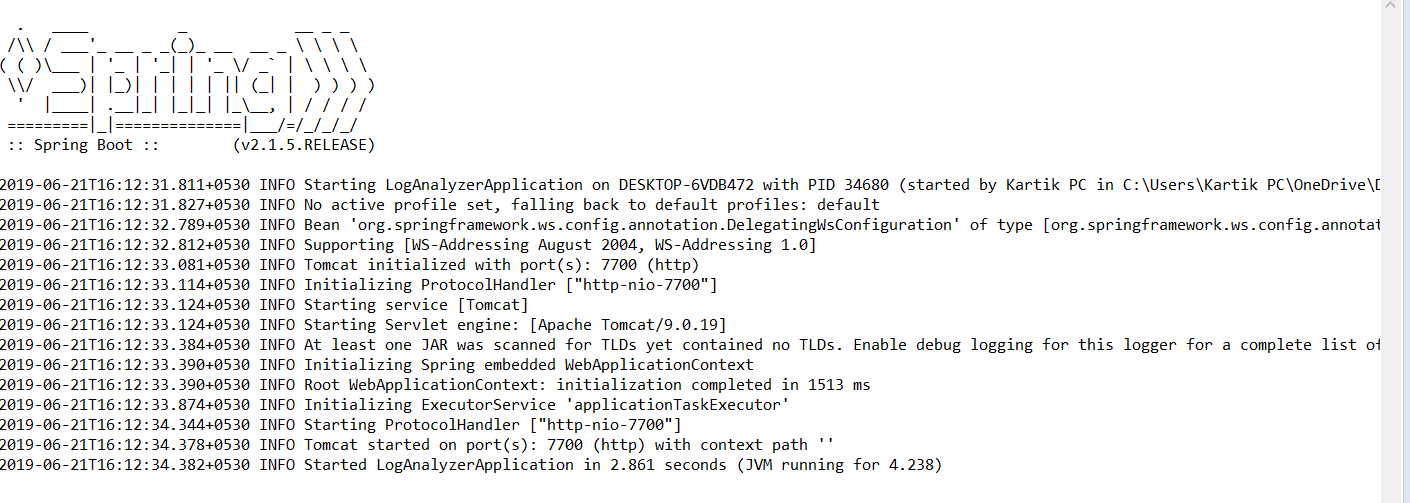
* A progress bar shows all the dependencies being installed



* Right click on project -> Run As -> Spring Boot App

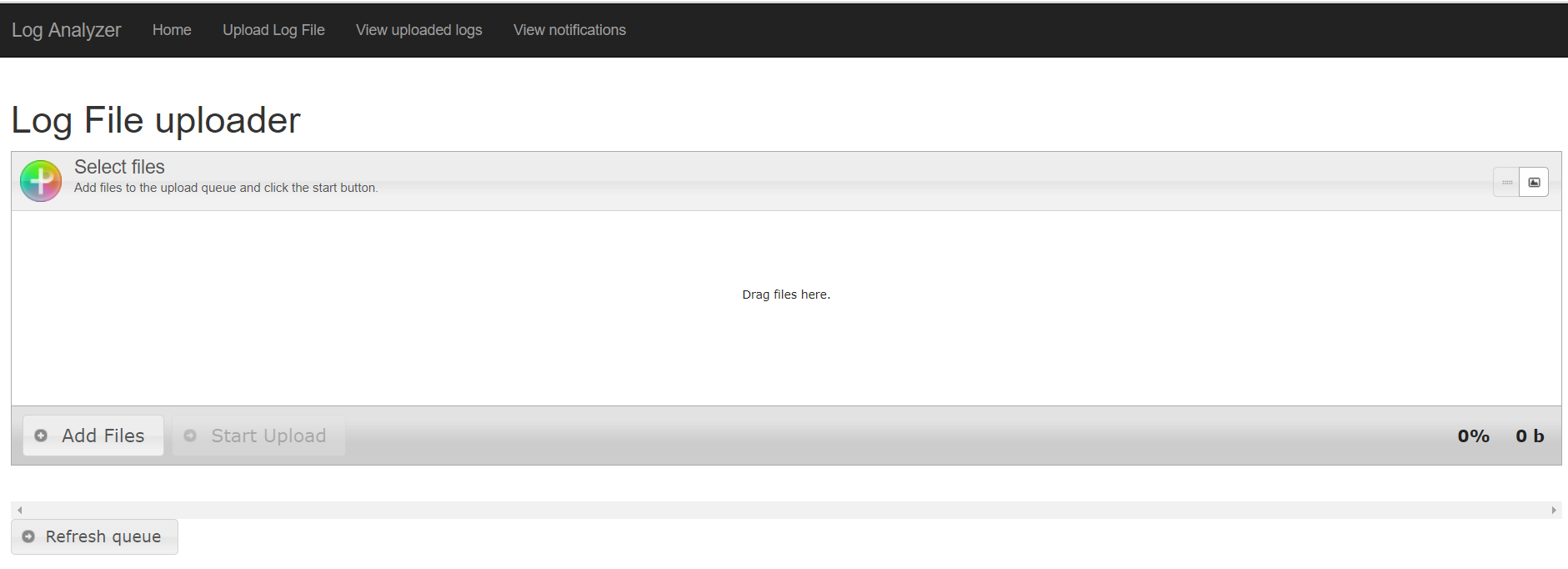


* When console shows this view, application has booted successfully

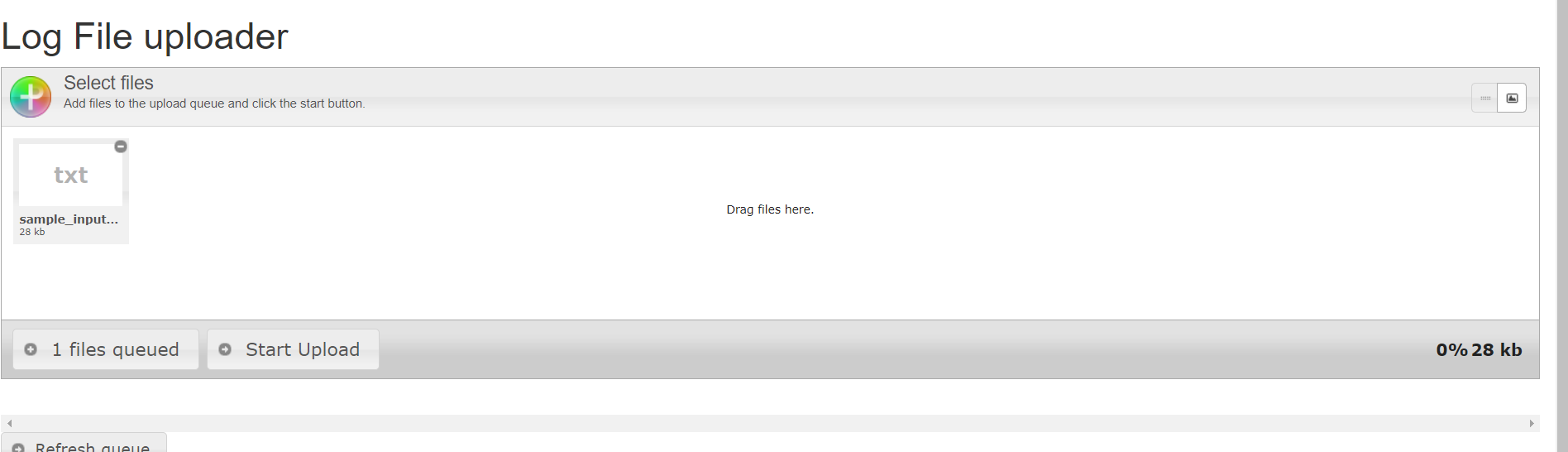


Using the application:

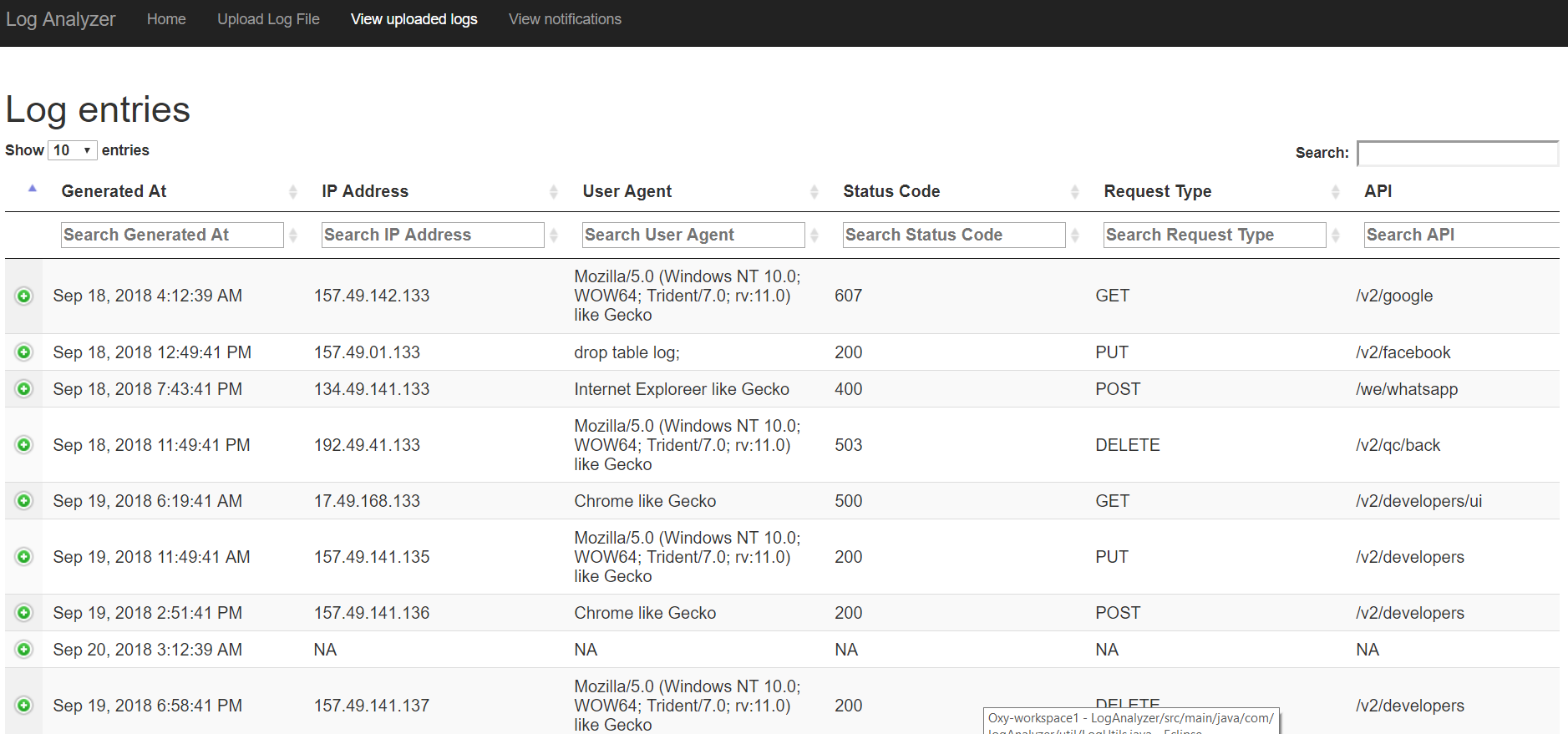
1. Open <http://localhost:7700/logAnalyzer/main>
2. Click ‘Upload File Queue’ to start uploading files



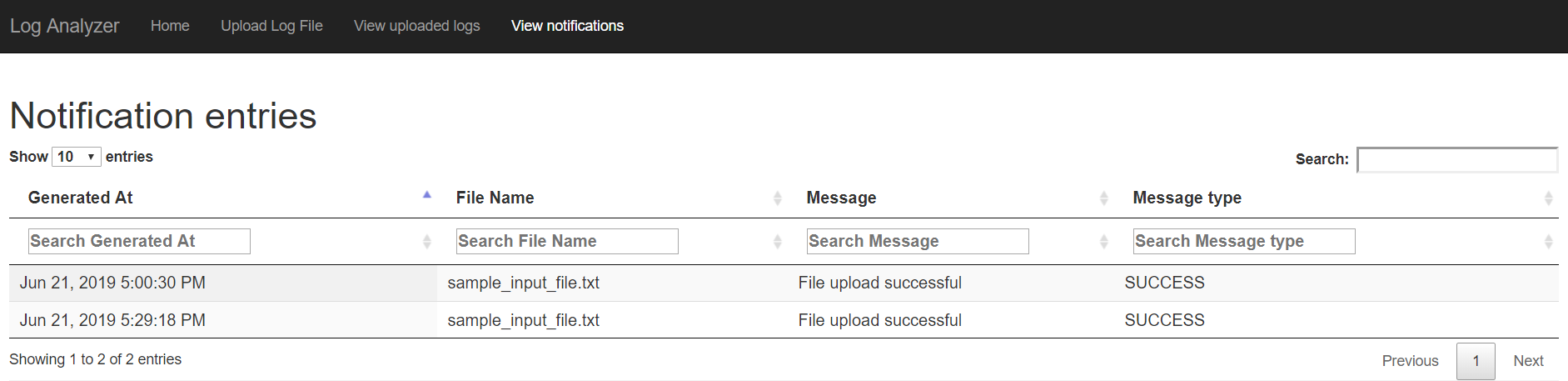
1. Click ‘Add files’ to upload
2. Click ‘Start Upload’ to start uploading



1. Click ‘View uploaded logs’ to view the already uploaded logs



1. Click ‘View notifications’ to get notifications list regarding the files uploaded



Approach:

1. File upload

Files to be uploaded can be large of the size 1 to 2gb. A normal HTTP request cannot handle this handle upload.

Plupload is a client library which sends the uploaded file in chunks to the server. These chunks are collected in a file and stored on the disk. While these chunks are being collected, the user is free to access the already uploaded logs and notifications. (non-blocking and asynchronous mechanism)

1. Analysing the log file

Once the entire file has been received, a JMS call is sent to create an asynchronous activity for file processing in the background. This processing activity is sent to ActiveMQ messaging queue with name “logFileQueue”. A JMSReceiver listens to this queue and executes the processing task. Meanwhile, if other files are uploaded, the processing tasks are pushed to this queue in the background. The java.nio package is used to fetch the file from the disk and create a Stream of lines. The file is processed line by line from this Stream. Each line of the file is examined using regular expressions for getting the following data:

* Timestamp
* Ip address
* User agent
* Status Code
* Request Type
* Api
* User Login
* User Name
* Enterprise Id
* Enterprise Name
* Entire log string

Data for all these lines is maintained in a List.

1. Inserting log data in MySQL database:

The list of lines created is inserted using batchUpdate of JdbcTemplate. batchUpdate helps to execute a number of queries in a ‘batch’ to the database. This saves the number of times the application connects to the database, thereby improving speed of insertion. Here, insert query is specified and the list is passed to this method. Batch size can also be specified in batchUpdate.

While logs are being inserted, user can see those logs by clicking on ‘View uploaded logs’.

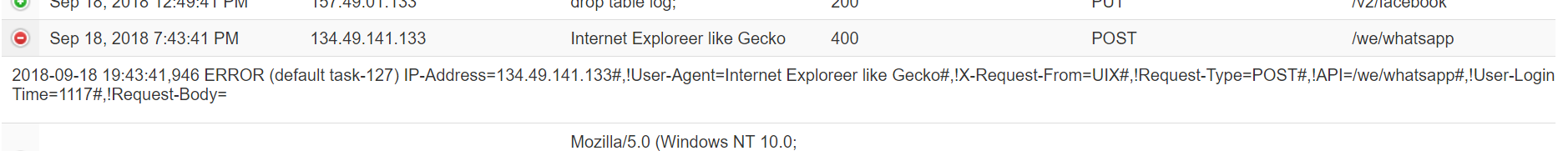
1. Notification

Once all records of the log file have been inserted in the database, a notification timeline log is generated in the database. This can be a success or error message depending on upload completion or validation failures respectively

1. Viewing the logs

View logs UI provides the list of logs. The amount of data processed will be huge. It is ensured that data will be paginated and only limited data will be available on each fetching of data. (server side processing)

There are some logs, which may not have the parameters being captured, but are still useful for analysis (like Exceptions). Such lines can be drilled down further by looking at the original log string. For this purpose, each log entry has a ‘+’ icon. Clicking this icon provides the log string.



1. Viewing the notifications

Notifications data will be fetched from server in paginated fashion, fetching only limited data on each request

Assumptions:

1. File to be uploaded:

* Only .txt files can be uploaded
* All the lines in the file which start with timestamp syntax mentioned in the work-sample document will be processed
* All the lines in the file mentioned in the second point should be exactly in the format mentioned in the work-sample document to get other details like Ip address, user details, request etc.
* If there are lines in the file which do not fulfil the third point but fulfil the second point (for eg. exception logs, stacktrace, etc.), the line will be processed by recording the timestamp and considering all other parameters as “NA”
* Lines which do not start with timestamp syntax will be skipped
* If some problem occurs with file processing, logs will not be stored in database, and an error notification will be recorded

1. Filtering:

This can be done among logs and notifications only through text boxes of individual columns

Possible improvements:

The application is still a work in progress towards perfection. There are a number of issues which may arise at present when it comes to performance. Following are some of the improvements possible and would have been implemented with some more time availability:

1. File processing is done line by line. It can be made event faster by processing it in a buffer fashion. java.nio file allows to receive specific number of bytes and store in a Buffer. Extra logic is needed to process the buffer and read each log line efficiently. This reduces the number of IO calls and increases speed. But it also adds the complexity of processing separate log lines. An approach which handles this trade-off has to written
2. The application is an attempt to create a single page application. However, it can be made even better by using JS frameworks like AngularJS, ReactJS, etc., which is currently not being used
3. ActiveMQ and JMS are still being used with default configurations. Their configuration can be modified to improve the performance even further
4. Filtering in dataTable is currently only allowed through text box. It can be made even better through dropdown, range filtering for dates, etc
5. Using WebSockets to implement live notifications to the user