Reporting tool

# Open Reports

## Objective

The purpose of this is for letting us build a proper tool useful to get reports on errors thrown by OGI’s applications during their runtimes. Since most of the “red ones” are ignored during development, we want to keep track of them all.

With this document, we want to get to know every single use that we can and want to take from this app.

### User stories

As a user, I want to get reports from every single application and eBroker site in one single report.

As a user, I want to filter the severity of the errors to be showed on the reports based on a list of all the severe types that are available.

## To do List

* Change from XML reading/writing to DB tables and stored procedures
* Change from Swing to Spring (?)
* Clear up the code to improve performance
* Try to write the code more JAVA 8 alike.
* Cache the data
* Make the similarity configurable. (Some of the work is already done in my home desktop)
* Change the result from plain text/XML data to tables to be sent to the destination developers’ emails.

### Things to notice

* Writing to an XML file to give some sort of intelligence to the tool
* Ability to merge different log messages to a common type message
* Treat different entries according to their types
* Use of JaroWinkler class from uk.ac.shef.wit.simmetrics.similaritymetrics

From the above list, the only point that requires a bit more of explanation is the last one. This JaroWinkler’s similarity metrics is used to compare how similar two string are. In this tool, I’m using so the XML/Database information doesn’t have to store similar data by checking if there’s a percentage of similarity between the line we’re reading from the logs to the existing ones in the XML/Database.

## Configuration

There’s an XML file with the “known”/most common log data and during the report this file is dynamically updated with all the new information found in the log file. Ideally, this **MUST** to be changed so it uses a database connection with the whole information – it improves the performance. There’s a small issue that I haven’t given a lot of thought about which is the case of existing very similar entries in the XML file, although I’ve implemented some intelligence to deal with it, it isn’t perfect yet. So the workaround I’ve been doing is to manually go to the XML file and change manually. This is an extra necessary workload for the time being, but I think it’s possible to “fix” this.

Apart from that, the current GUI deals with the report data we want to get.

### Add new applications

To add new applications to the reporter, you’ll have to go to:

* Create a new class inside lib.gui.applications.specific package named as **\*AppName\*ApplicationPanel.java** which extends ApplicationPanel.java (follow the existing ones to match consistency)
* **ApplicationsPanel.java**: add it in the *private static final String \*appName\*Application* section and then add it a new switch case inside addNewApplicationPanel using the previously created class.
* **OpenReportConfig.java:** add it wherever you find appropriate (for now it is in alphabetical order)
* **ReportConfig.java:** add it inside the init() method as **this.applications.put(ApplicationsPanel.\*appName\*Application, null);**

### Others

As for the configuration, the information already given must be enough, however, apart from the A*pplications* panel, all the others are very easy to update. There are some loose strings that can easily be refactored and improved.

## Screenshots

Figure 1 - Main panel GUI

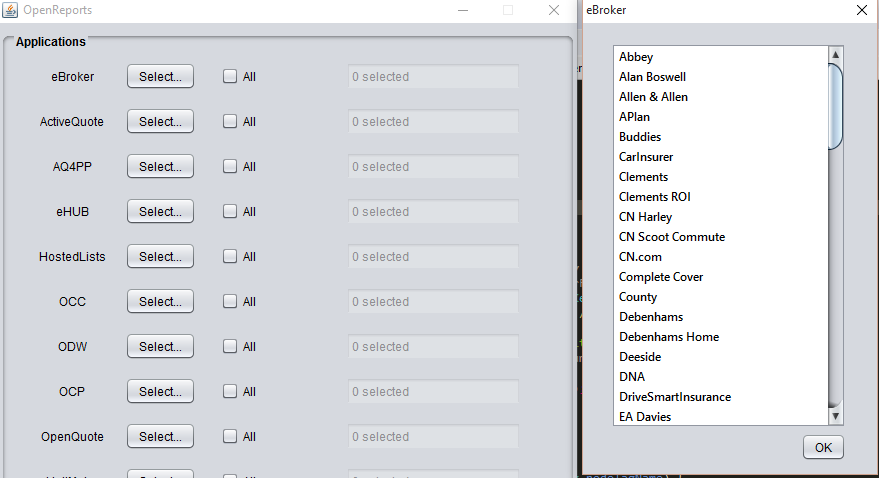


Figure 2 - Application Details

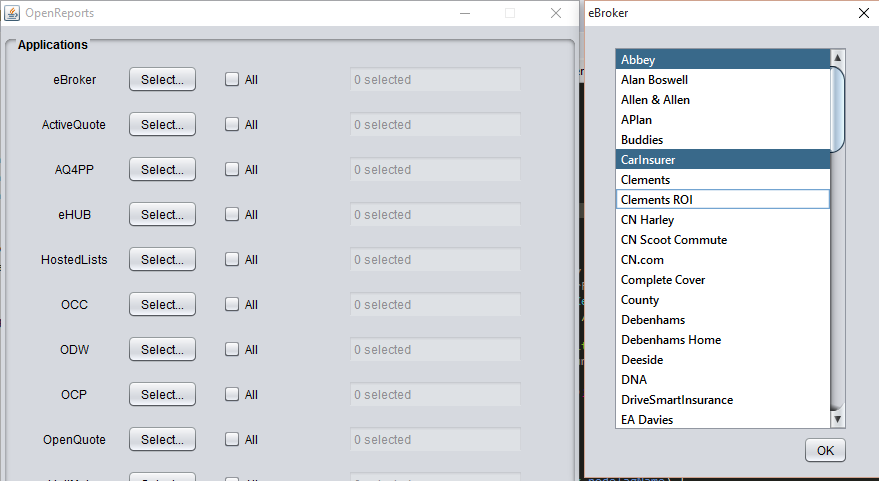


Figure 3 - Application Details

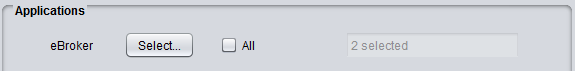


Figure 4 - When you click "All", all the servers are selected

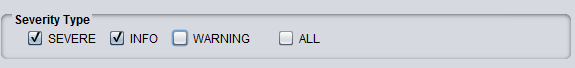


Figure 5 - Filter the severity level

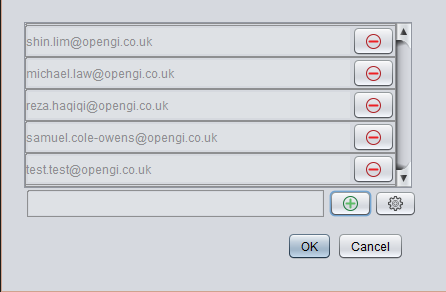
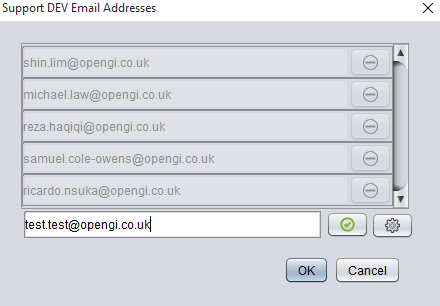
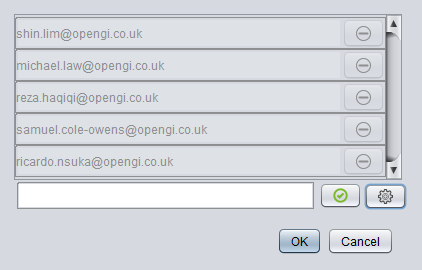
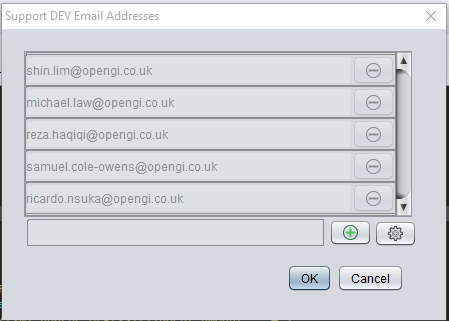


Figure 6 - Support developers list



Figure 7 - Button to run the report

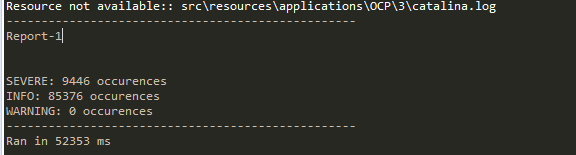


Figure 8 - Information relative to the report result

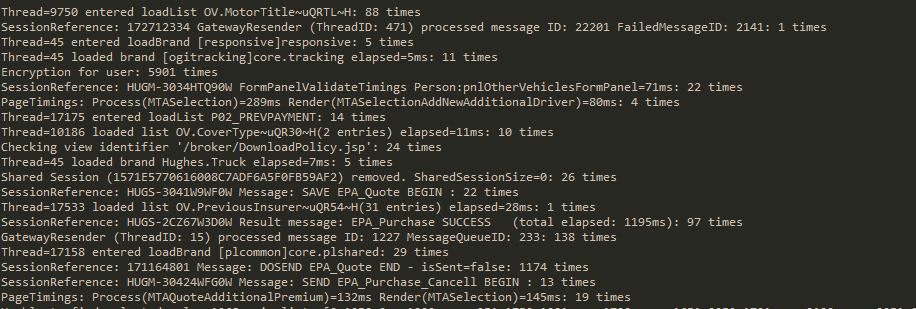


Figure 9 - How many times the messages appeared in the logs

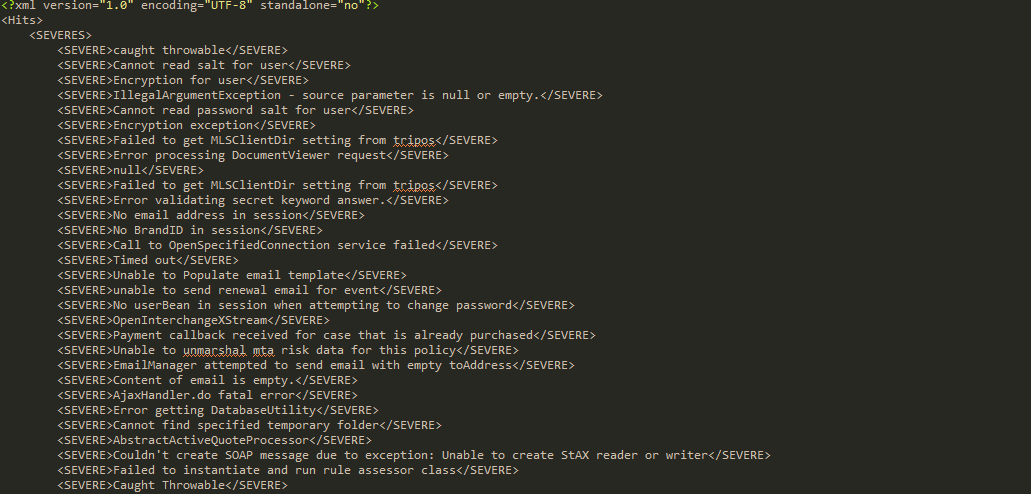


Figure 10 - XML file with the known log information