JMeter Automation Framework

Author: Uma Mahesh

Date: 26/03/2015

# Goal:

JMeter automation framework goal to automate the Load Performance tests using JMeter in command line and unattended mode & uploads results to the My SQL database and integrating with Go Server for Continuous Integration purpose.

The JMeter Automation framework contains the following components.

* 1. MySQL Schema
  2. Framework Structure
  3. JMeter Command line

# Database Schema

The following tables have been created for updating JMeter results in the database.

1. Perf\_run\_details
2. Perf\_run\_results
3. Perf\_run\_requests\_summary
4. Perf\_run\_errors

Perf\_run\_results, Perf\_run\_requests\_summary, Perf\_run\_errors tables are dependent on the **Perf\_run\_details table ID** column primary key and it is reference to the rest of the tables as a foreign key.

**Performance Run Details:** This table is mainly to create JMeter run jobs and details in an incremental way and to identify each run with unique identification.

Id (Primary Key), description, date and time the run started, status of the run

CREATE TABLE `perf\_run\_details` ( `id` int(11) NOT NULL AUTO\_INCREMENT, `build\_tag` varchar(45) DEFAULT NULL, `description` varchar(250) DEFAULT NULL, `date\_time` datetime DEFAULT NULL, `status` varchar(45) DEFAULT NULL, PRIMARY KEY (`id`)) ENGINE=InnoDB AUTO\_INCREMENT=53 DEFAULT CHARSET=utf8;

**Performance Run Results:** This table is mainly to update the JMeter results for the jtl generated at the time of command line and converted to csv file. This will get updated in the database once test completes.

CREATE TABLE `perf\_run\_results` (

`agg\_sampler\_name` varchar(250) NOT NULL,

`agg\_no\_of\_requests` int(11) DEFAULT NULL,

`agg\_avg\_response\_time\_ms` int(11) DEFAULT NULL,

`agg\_report\_median` int(11) DEFAULT NULL,

`agg\_report\_90%\_line` int(11) DEFAULT NULL,

`agg\_report\_min\_ms` int(11) DEFAULT NULL,

`agg\_report\_max\_ms` int(11) DEFAULT NULL,

`agg\_report\_error\_percent` varchar(100) DEFAULT NULL,

`agg\_report\_throughput` double DEFAULT NULL,

`agg\_report\_throughput\_kbs)` double DEFAULT NULL,

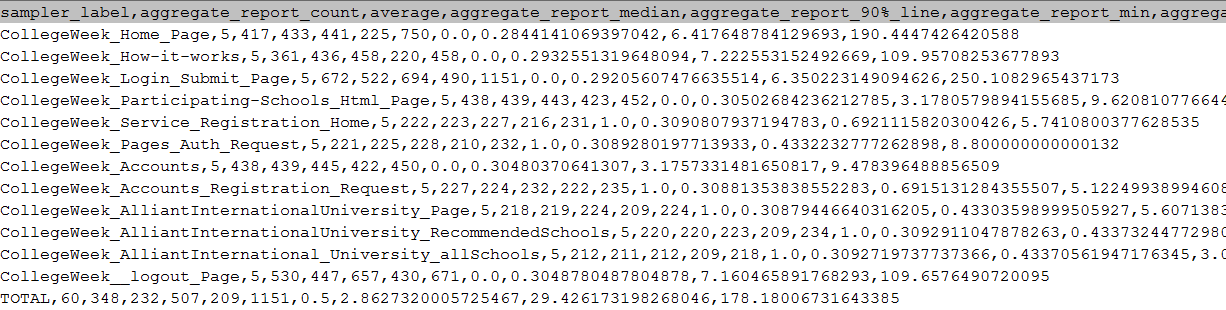
`agg\_report\_stddev` double DEFAULT NULL,

`id` int(11) DEFAULT NULL,

KEY `FK\_Perf\_Runs\_ID\_id` (`id`),

CONSTRAINT `Perf\_Runs\_ID\_FK` FOREIGN KEY (`id`) REFERENCES `perf\_run\_details` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;



**Performance Requests Results:** This table is mainly to update the JMeter each requests details results in to the database. This table gives detail report about each requests response time, date time, response code, response message and no of active users and total users.

CREATE TABLE `perf\_run\_requests\_summary` (

`detail\_date\_time` timestamp NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE CURRENT\_TIMESTAMP,

`detail\_sampler\_name` varchar(250) DEFAULT NULL,

`detail\_response\_time` int(25) DEFAULT NULL,

`detail\_http\_return\_code` varchar(500) DEFAULT NULL,

`detail\_status\_message` varchar(800) DEFAULT NULL,

`detail\_thread\_group\_name` varchar(250) DEFAULT NULL,

`detail\_message\_type` varchar(250) DEFAULT NULL,

`detail\_condition` varchar(45) DEFAULT NULL,

`detail\_Bytes\_kb` int(50) DEFAULT NULL,

`detail\_active\_threads` int(25) DEFAULT NULL,

`detail\_no\_of\_threads` int(25) DEFAULT NULL,

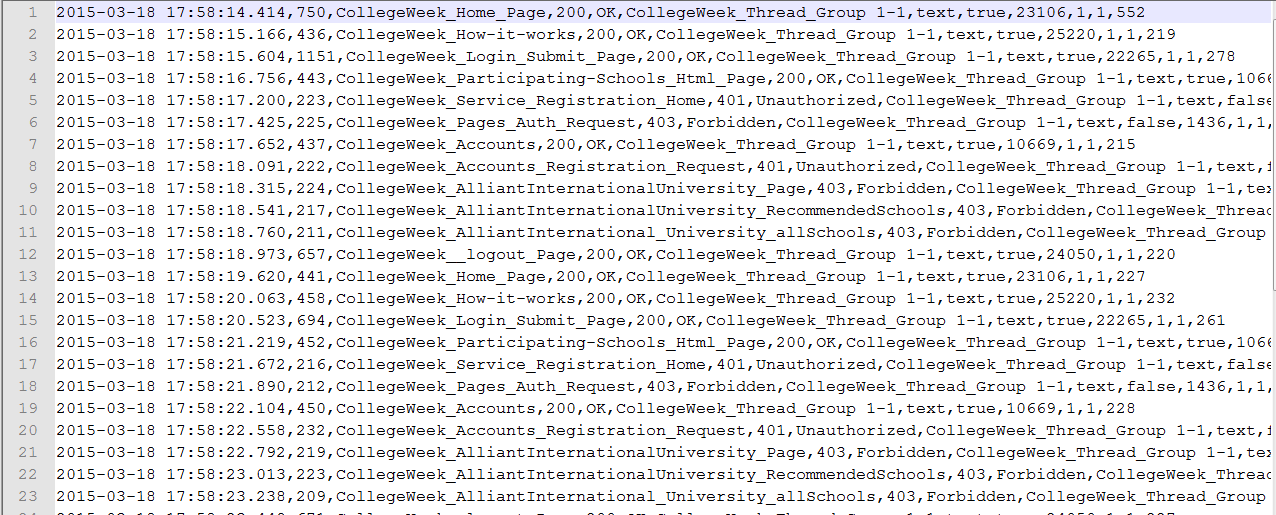
`detail\_latency` int(25) DEFAULT NULL,

`id` int(11) DEFAULT NULL,

KEY `Perf\_Runs\_detail\_FK\_id` (`id`),

CONSTRAINT `Perf\_Runs\_detail\_FK` FOREIGN KEY (`id`) REFERENCES `perf\_run\_details` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;



**Performance Errors Results:** This table is mainly to update the JMeter requests failures in the database to understand the failures in the requests.

CREATE TABLE `perf\_run\_errors` (

`date\_time` datetime NOT NULL DEFAULT '0000-00-00 00:00:00' ON UPDATE CURRENT\_TIMESTAMP,

`elapsed\_time` varchar(45) DEFAULT NULL,

`sampler\_name` varchar(500) DEFAULT NULL,

`response\_code` varchar(500) DEFAULT NULL,

`response\_message` varchar(800) DEFAULT NULL,

`thread\_name` varchar(500) DEFAULT NULL,

`data\_type` varchar(45) DEFAULT NULL,

`success\_code` varchar(45) DEFAULT NULL,

`bytes` int(11) DEFAULT NULL,

`group\_threads` int(11) DEFAULT NULL,

`num\_threads` int(11) DEFAULT NULL,

`latency` int(11) DEFAULT NULL,

`sample\_count` int(11) DEFAULT NULL,

`error\_count` int(11) DEFAULT NULL,

`id` int(11) DEFAULT NULL,

KEY `FK\_Perf\_Runs\_Errors\_idx` (`id`),

CONSTRAINT `FK\_Perf\_Runs\_Errors` FOREIGN KEY (`id`) REFERENCES `perf\_run\_details` (`id`) ON DELETE NO ACTION ON UPDATE NO ACTION

) ENGINE=InnoDB DEFAULT CHARSET=utf8;

# productname : Name of the Product

productname=PlatformQ

# testplanname : Name of the JMX file or Scenario Name

testplanname=LoadTestDev

# releasenum : Release number

releasenum=1.0

# buildNum : Build Number

buildNum=#123

#documentTitle=Title of the Test

documentTitle=Jenkins\_PerformanceTest

#Report Parameters

#To skip the tests and to generate only reports.

skipTests=false

#To skip the reports and run only the tests.

skipReports=false

#To skip only aggreagte reports.

skipAggregateReports=false

#OS Parameters

# The parameter to choose the jmeter.bat or sh based on the operating system.

OS=Linux

#JmeterLinuxDir

# Don't change the below results directories

resultsdir=/code/results

jmeterdir=/code/jmeter

resultsmasterdir=/masterresults

# Give the test jmx file and it's related folder to run the test

jmxdir=/code/jmx/LoadTestDev

#Database details

# database IPAddress

dbipaddress=misc.cgfr0rj0lp3w.us-east-1.rds.amazonaws.com

# database SchemaName

dbschemaname=cwlTestResults

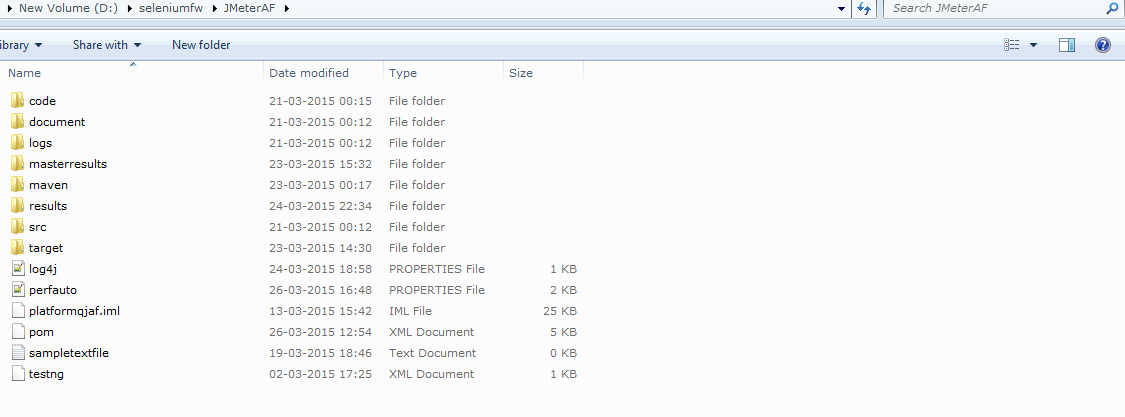
# Database username and Password

dbusername=testresults

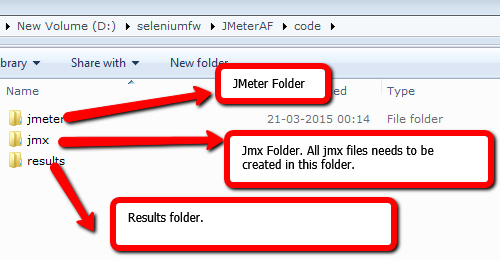
dbpassword=DSwVVYJt

# Framework Structure

The JMeterAF contains folder structure as shown in the below snapshot.

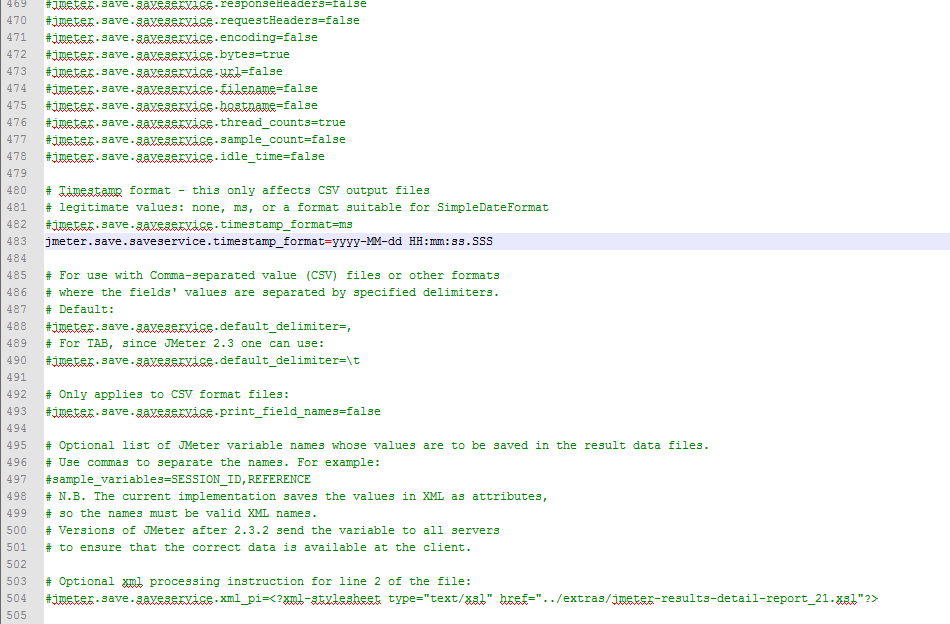


The code folder contains Jmeter, jmx and results folders.

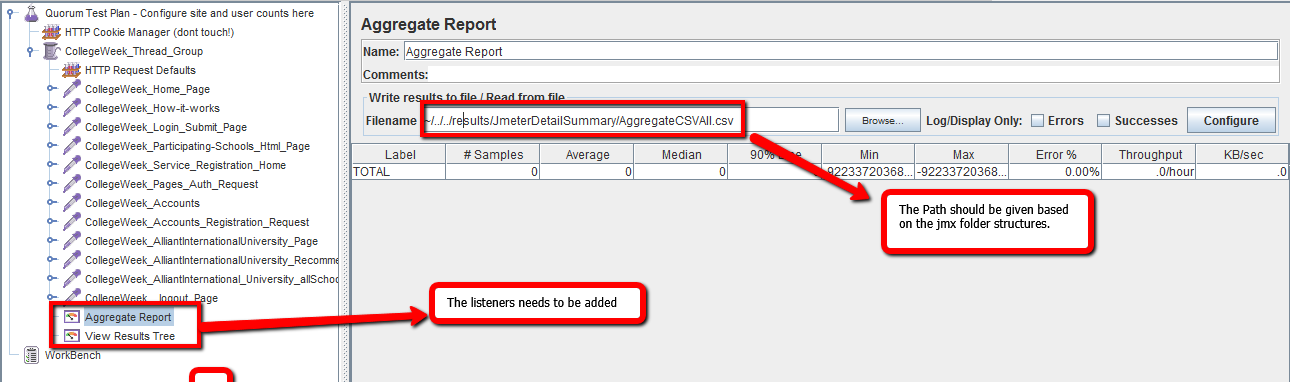


The following Changes need to be done to get the all reports in the database.

1. JMeter properties file : The time needs be changed as shown in the screenshot



1. View Results and aggregate report listener’s needs to be added to the JMX files. And the following options needs to be checked for the jtl results.



**Results File Name : ~/../../results/JmeterDetailSummary/AggregateCSVAll.csv**

**Ex: -** D:\seleniumfw\JMeterAF\code\jmx\LoadTestDev -- If the JMX is in LoadTestDev folder then the results path should be given as below.

**~/../../results/ JmeterDetailSummary/AggregateCSVAll.csv**

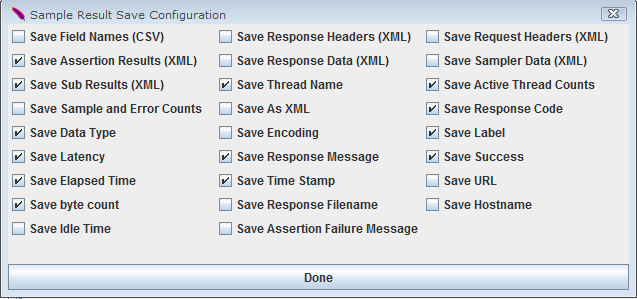
**D:\seleniumfw\JMeterAF\code is nothing but a root folder = ~/**

**D:\seleniumfw\JMeterAF\code\jmx is nothing but =~/../**

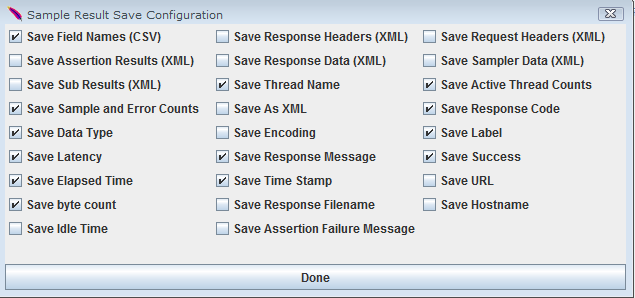
**D:\seleniumfw\JMeterAF\code\jmx\LoadTestDev is nothing but =~/../../**

If you are keeping any jmx file below LoadTestDev folder with folder name example (D:\seleniumfw\JMeterAF\code\jmx\LoadTestDev\example) then we have to mention the results path as ~/../../../results/ JmeterDetailSummary/AggregateCSVAll.csv

**Aggregate Report:**



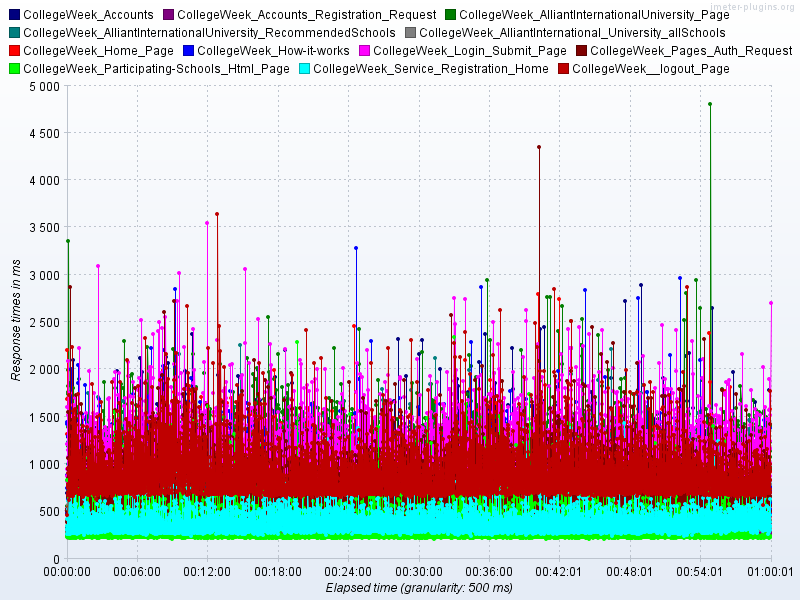
**View Results Tree:**

****

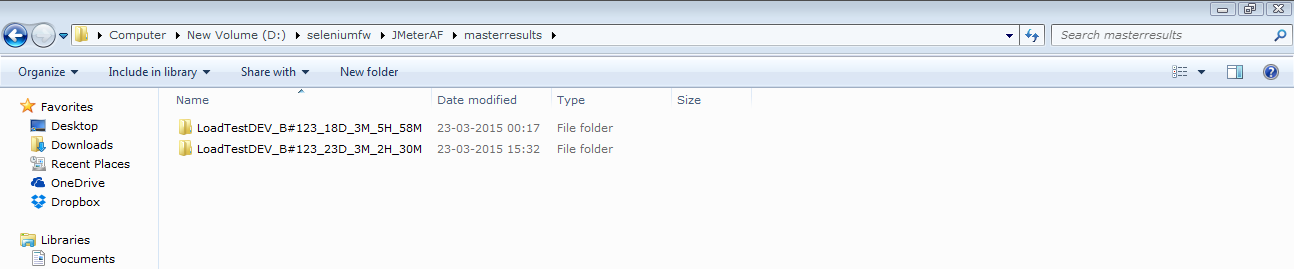
# Run a Test

1. To run test just give the “jmxdir” folder name in the perfauto.properties file.
2. Go to command line option and run the following command to execute the test.
   1. mvn clean install pom.xml
   2. mvn test –f pom.xml
3. The test will get started and it will run the test based on the jmx file present in the mentioned directory.
   1. If the directory has multiple jmx files framework will run all the jmx files
   2. If the directory has jmx and directory files then it will run jmx files in the given directory and also from the child directories.
4. Creates a performance job details in the **Perf\_run\_details** table by creating a id.
5. Once the run completes it generates the following reports and updates results, details results and errors in to the database tables and updates the status to passed or failed.
   1. AggregateReport
   2. HitsPerSecond
   3. ResponseTimesOverTime
   4. TransactionsPerSecond
   5. BytesThroughputOverTime
   6. LatenciesOverTime
   7. ThroughputVsThreads
   8. ThreadsStateOverTime
   9. ResponseCodesPerSecond

**Sample Report:**

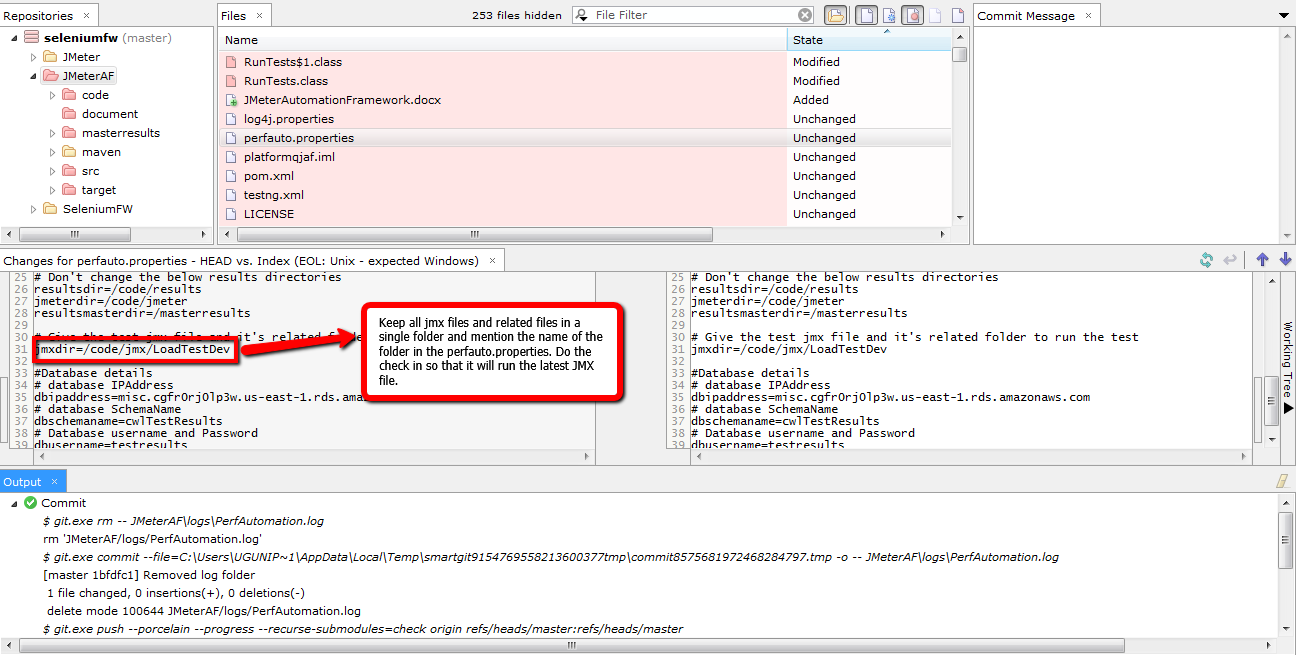


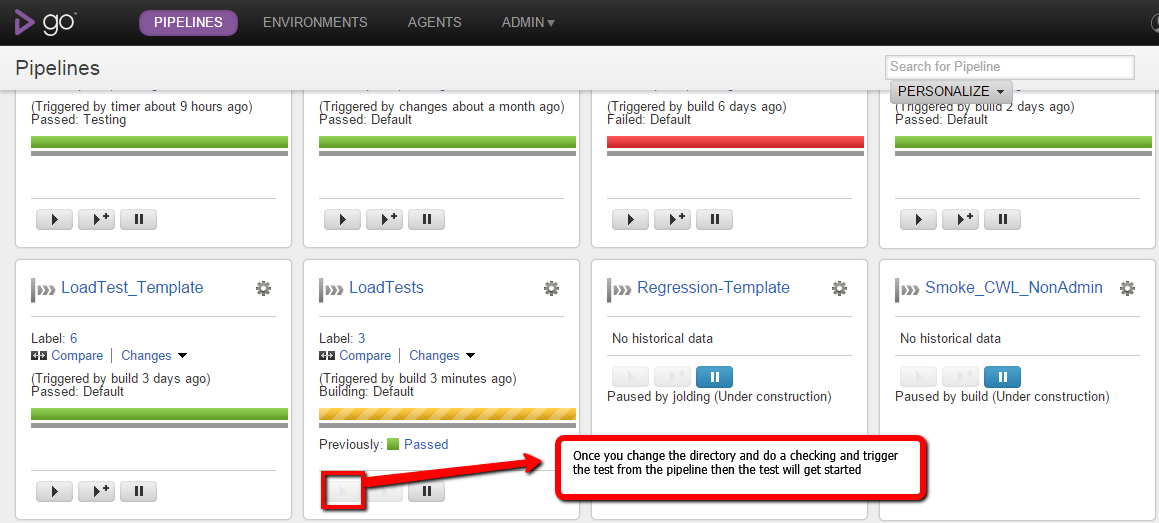
6. Finally copies all the results into MasterResults folder.



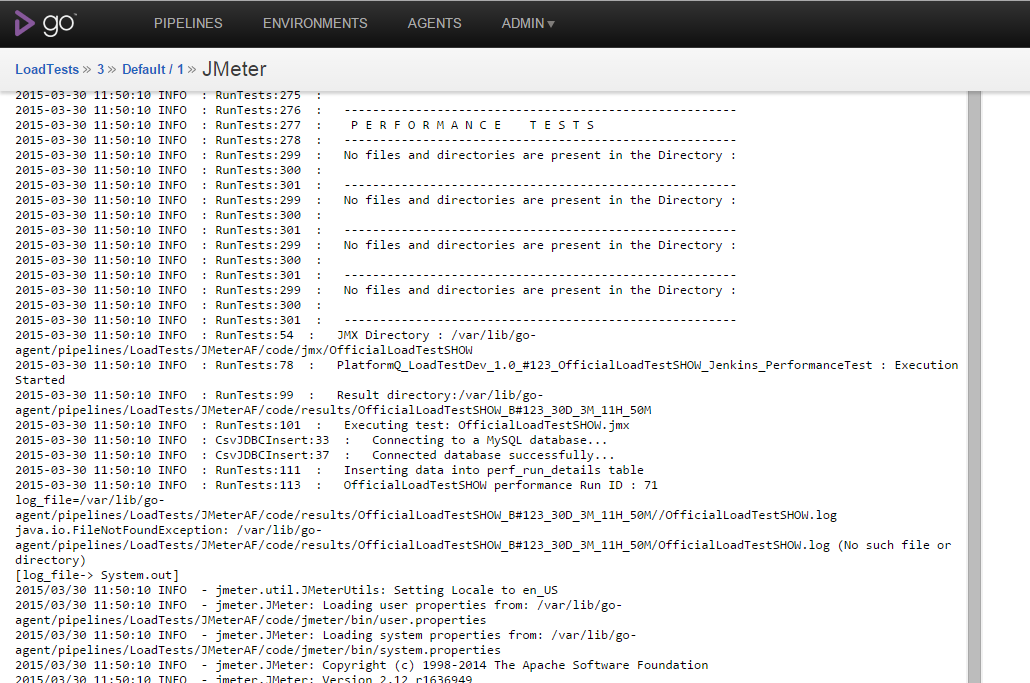
# Go Server:

1. Go to <http://thoughtworks-go-lb-196250922.us-east-1.elb.amazonaws.com/go/tab/pipeline/history/LoadTests>
2. Before running a test on the Go Server just update the Jmx folder name in the perfauto.properties



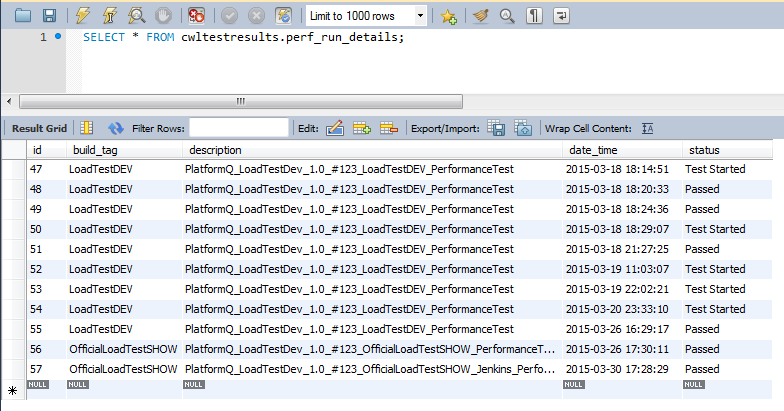


1. Go to the Job Console and check for the log to see the run details

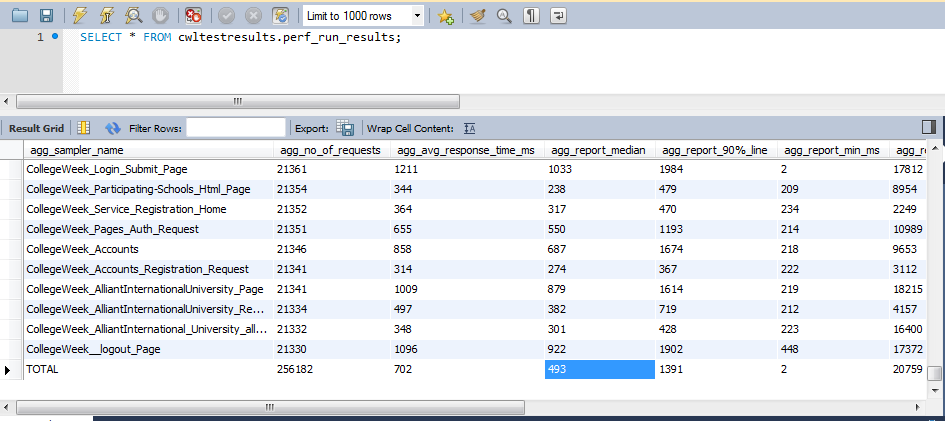


1. Once it is completed then you can see the results in the database.

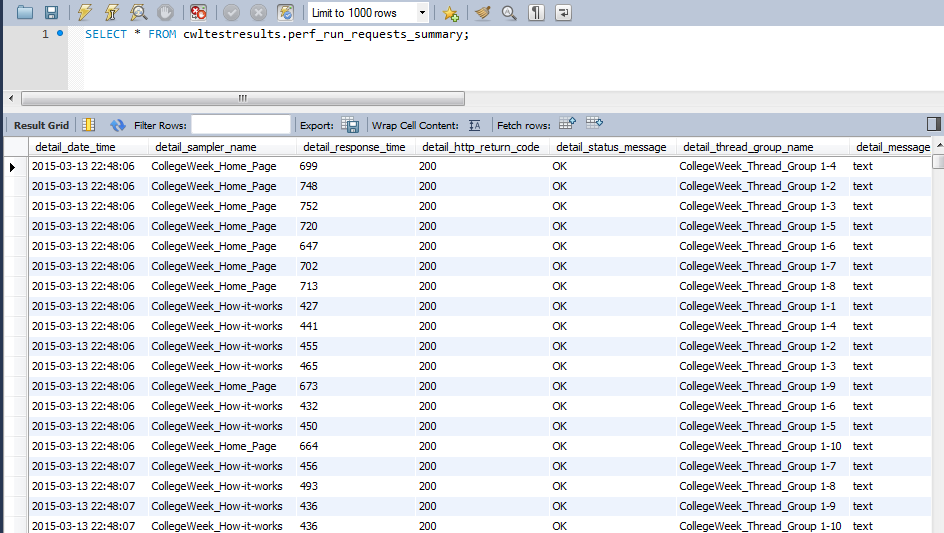
**Run Details**



**Run\_Results**



**Requests\_Summary**



**Errors:**

