Software Design Document

For

Enhancements to JMeter

Requirements for JMeter version 2.9 and above

Prepared by JMeter Group, Summer Interns

IIT Bombay

25/06/2013

Table of Contents

Team Members

Buddha Sushmitha (KAKINDA INISTITUTE OF ENGINEERING AND TECHNOLOGY): Worked on Jmeter plugins and Jmeter graphs.

Dhruv Joshi (NIT, Rourkela): Worked on Filter Results Plugin, Listeners and tested Long Polling technology implemented on Moodle using JMeter

Manisha Choudhury(NIT, Rourkela): Worked on Samplers and auto csv creation.

Naman Choudhary(NIT, Jamshedpur): Worked on Controllers, Preprocessors, Object Diagrams and Automating TPC-C.

Shekhar Saurav (NIT, Jamshedpur): Worked on Config elements and Dyanmic Bandwidth Throttling.

Surabhi Mour (NIT, Surat): Worked on Assertions, Post-processors and Automating TPC-C.

1 INTRODUCTION

1.1 Purpose

The purpose of this document is to specify the requirements and preview some elements of the Load Testing tool JMeter.

Apache JMeter desktop application is open source software. The application designed to load test functional behavior and measure performance. It was originally designed for testing Web Applications but has since expanded to other test functions.

Various Experiments are to be conducted using JMeter. Variety of applications can be tested and graded using JMeter, bottlenecks can be identified by varying parameters of each experiment. JMeter can load test various types of servers like:

- Web - HTTP, HTTPS
- SOAP
- Database via JDBC
- LDAP
- JMS
- Mail - SMTP(S), POP3(S) and IMAP(S)
- Native commands or shell scripts

JMeter is highly extensible, many open source plugins are available for JMeter for each of its major component like samplers, listeners, thread group, etc.

1.2 Scope

Apache JMeter is a Java desktop application designed to load test client/server software (such as a web application). It may be used to test performance both on static and dynamic resources such as static files, Java Servlets, CGI scripts, Java objects, databases, FTP servers, and more. JMeter can be used to simulate a heavy load on a server, network or object to test its strength or to analyze overall performance under different load types.

Additionally, JMeter can help you regression test your application by letting you create test scripts with assertions to validate that your application is returning the results you expect. For maximum flexibility, JMeter lets you create these assertions using regular expressions.

Current JMeter application has the robustness of testing various types of servers and also perform various types of testing, such as Load testing, Regression Testing, Functional Testing, Stress Testing, etc.

The new features introduced in JMeter will make the tool efficient for many other types of test scenarios which can introduce more practicality into the test scripts and user friendliness.

1.3 Overview

i) System Overview: Describes the system background and working
ii) System Architecture: Describes the software architecture followed by the system and detailed description of each component.
iii) Component Design: The logistics involved in designing each component that is to be developed.
iv) Human Interface Design: Detailed description of Interface of each component

1.4 Reference Material

[1] http://jmeter.apache.org/
[2] "Apache JMeter: A practical beginner's guide to automated testing and performance measurement for your websites"- Emily H. Hallili
[3] http://jmeter.apache.org/usermanual/component_reference.html#introduction
[4] http://www.code.google.com/p/jmeter-plugins/
[5] http://blackanvil.blogspot.in/2006/06/shootout-load-runner-vs-grinder-vs.html
[6] http://shantonusarker.blogspot.in/2013/05/introdution-to-jmeter-google-plugin.html
[7] http://www.methodsandtools.com/tools/jmeterplugins.php/

1.5 Definitions and Acronyms

Test plan: A test plan describes a series of steps JMeter will execute when run. Test plan is saved
Workbench: work bench is not saved with the test plan
CSV files: Common Separated Values
Jmx: Java Management Extension
Xml: Extensible Markup Language
TPC: Transaction Processing Performance Council
Plug-in: It is a software component

2 SYSTEM OVERVIEW

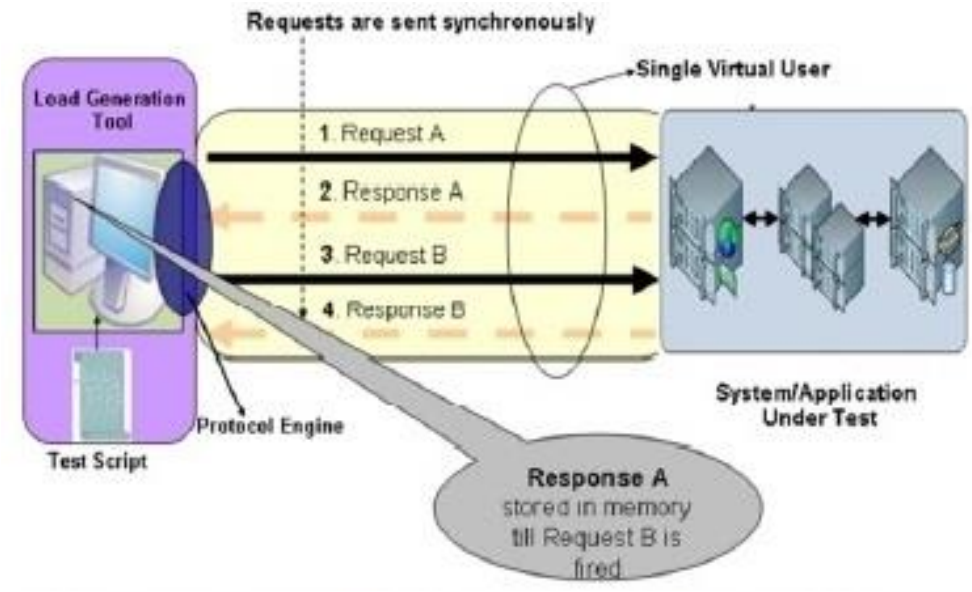
JMeter is an open source Apache Jakarta application. Used as a load testing tool for analyzing and measuring the performance of a variety of services, with a focus on web applications. Now can be extended to run the protocols Web (HTTP/HTTPS),FTP, JDBC, JMS, LDAP,SOAP. Originally developed by Stefano Mazzocchi.

3 SYSTEM ARCHITECTURE

Thread Group: This is the most important component in architecture of JMeter.
Thread: There can be single or multiple threads running in JMeter
Pre-processor: (*optional*) this does pre-processing on the sampler
Sampler: This component sends requests in the test to the Application Under Test
Post-processor: (*optional*) this performs post-processing on the sample results

Assertions: (*optional*) this asserts the results produced in samplers
Listeners: This is used to draw graphs or fill tables containing data from tests

3.1 Architectural Design



3.2 Decomposition Description

3.2.1 Assertions

Assertions are used to verify that the response of any samplers satisfies certain criteria. If the criteria specified is met, it is a passed assertion, else the assertion is said to have failed. This component is highly used in functional testing to check the responses received. Perl style regular expressions can be used to check the response data for some specified content. For e.g. a “welcome” can be checked after a login window to check for successful login attempt. Assertions can be specified to be applied to either only the samples to which they are added or to samples as well as sub-samples.

3.2.2 Config Elements

Config Elements or Configuration elements are used to set defaults and variables to be used by the samplers defined under their scope. There are 18 different config elements each for specific purpose as described below.

3.2.3 Logic Controllers

Logic Controllers determine the order in which Samplers are processed. As in an If controller allows the execution of the samplers within it only when the condition specified evaluates to true . Similarly the once only controller allows the execution of the samplers present within it only once per iteration.

3.2.4 Pre-processors

Preprocessors are used to modify the Samplers in their scope. There are nine types of preprocessors defined in JMeter.

3.2.5 POST-PROCESSORS

Post-processors execute after a request has been made from a Sampler. A good way is to place them as a child of a Sampler, to ensure that it runs only after a particular Sampler, not to Sampler afterwards. This element is most often used to process the response data, for example, to retrieve particular value for later use.

3.2.6 Timers

Timers are used to produce a particular amount of delay between the executions of the different threads. Timers are processed before each sampler in the scope in which they are found; if there are several timers in the same scope, all the timers will be processed before each sampler.

3.2.7 Samplers

Samplers are the most important component of JMeter, as they send requests to the Application Under Test and the results decide the performance of AUT

3.2.8 Listeners

Listeners are an integral component of any test plan, without Listeners one cannot analyze the results of a test. The basic Listener "Simple Data Writer" records all the data in CSV or XML format and stores it for suther reference. The Simple data writer is preferably used in non-GUI mode, as it saves the overhead of GUI functionality.
Listeners are prepped at the bottom of the scope in which they are kept.
Listeners can use a tonnes of memory space if the number of samples is huge.

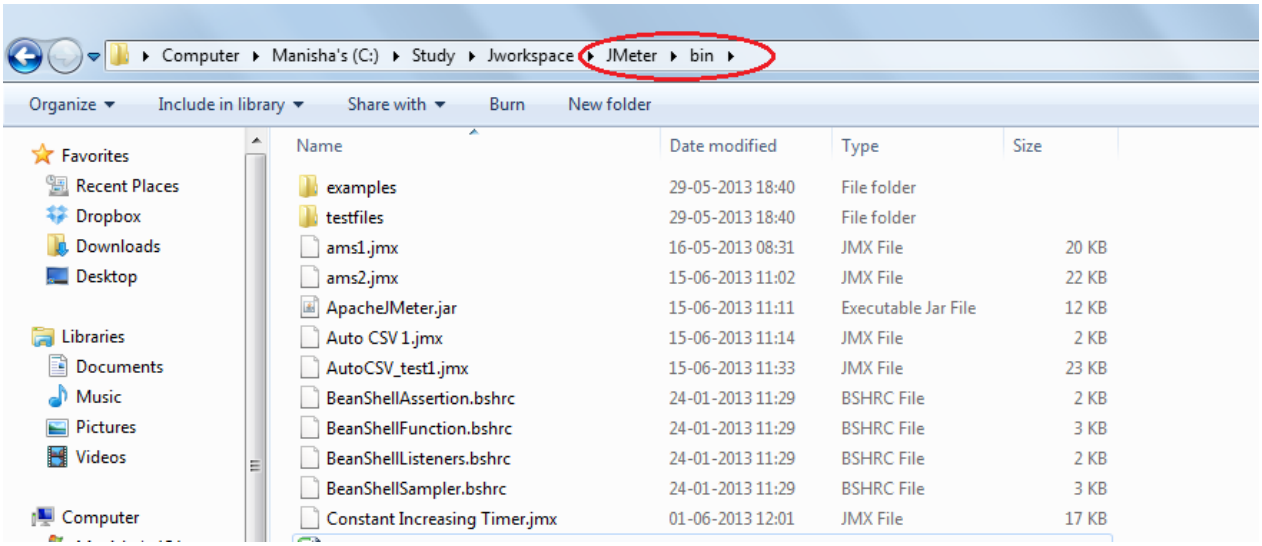
4 COMPONENT DESIGN

4.1 Test Plan to demonstrate "Auto CSV Generation" Configuration Element.

Aim: The aim of running this test is to automatically create the .csv file for the table of database mentioned, using "Auto CSV Generation" and to use it along with samplers to supply data to samples under use in test.

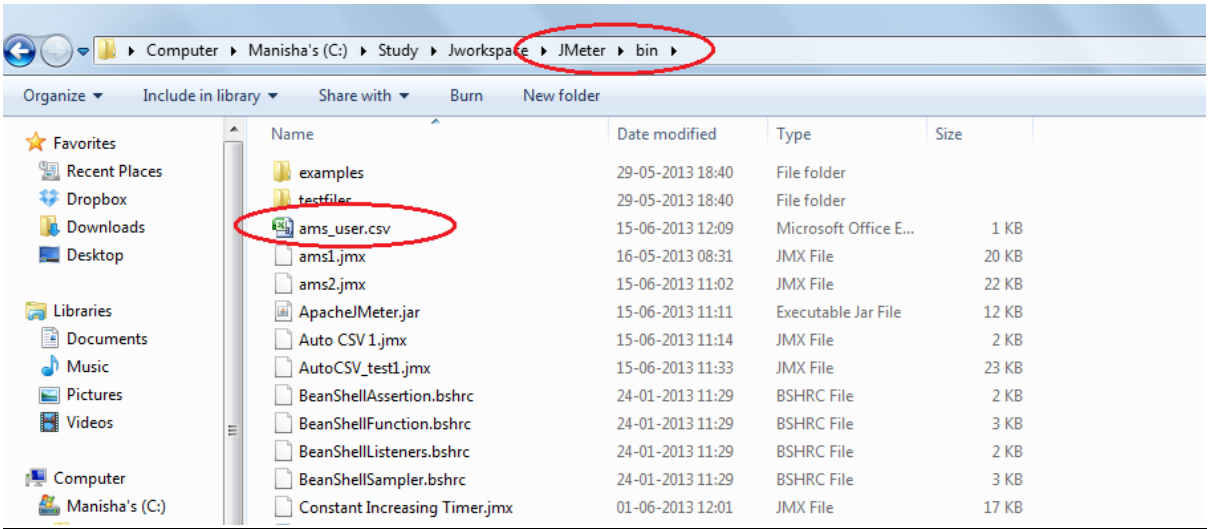
Procedure: The main purpose of the "Auto CSV Generation" is to create .csv file and have it in the "bin" folder of Jmeter to be used by other samplers.

View1: bin before csv file generation



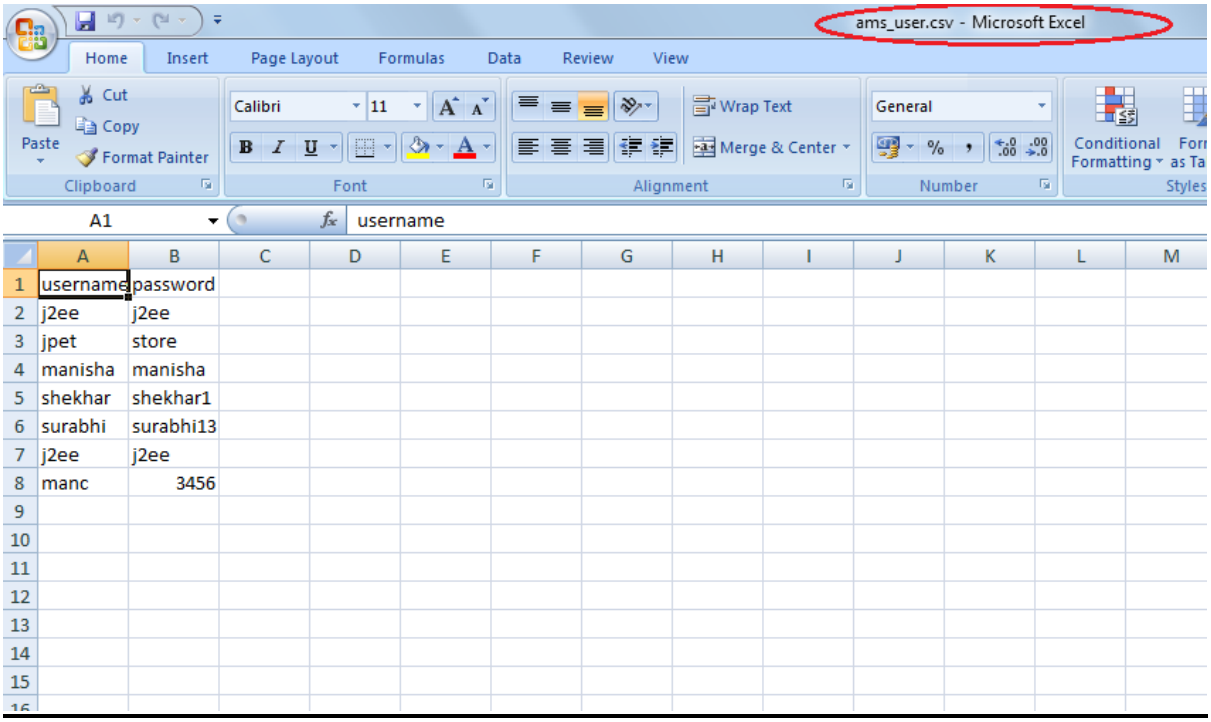
The database name and table name, along with connection details, is mentioned in the interface, and the new csv file will be named as databaseName_tableName.csv, and can be found in the bin of Jmeter.

View2: bin after csv file generation



The .csv file, created during a Jmeter experiment with Database –ams and Table –user, has the following content.

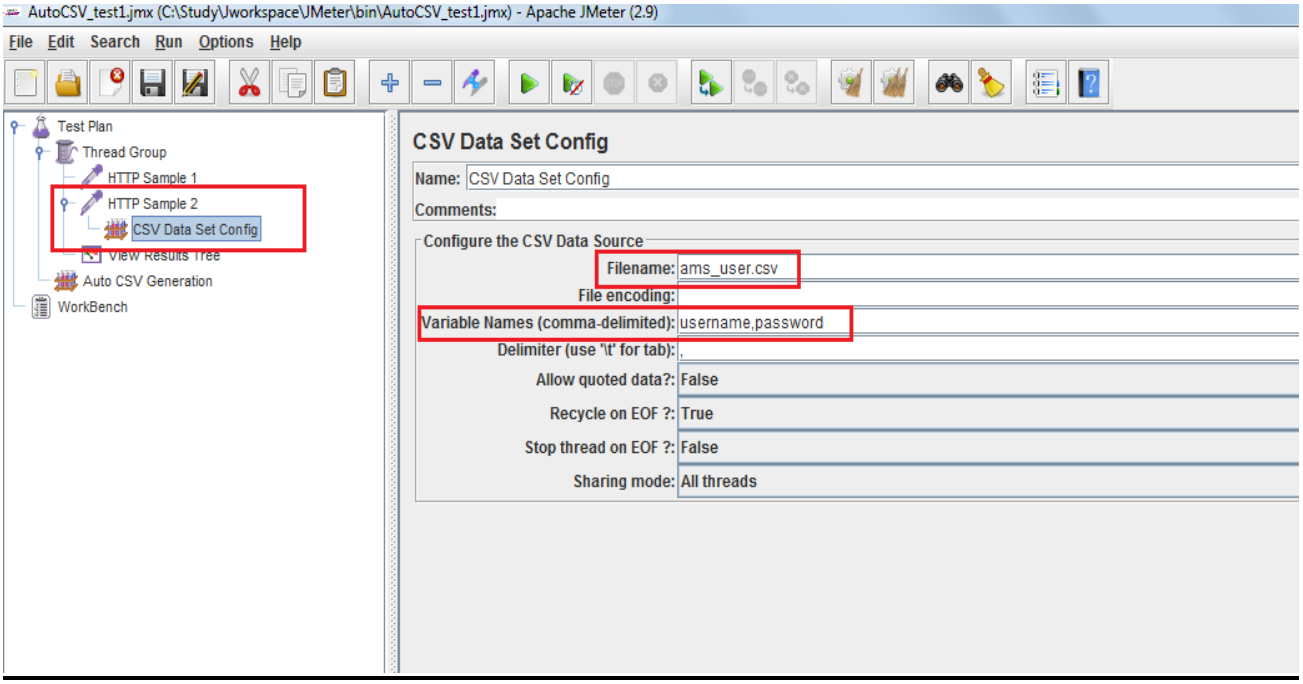
View3: The ams_user.csv file in MS Excel



The application under test (AUT) is a simple web-app, for airport management people, the opening page of which expects user name and password as user (manager) input, checks against Database already present and then creates session for users. Thus here the parameters passed are username and password, which need to be unique for all user logged in simultaneously. For a web-server testing, with load of 100 or 1000 users, a csv file needs to be produced with 100 or 1000 entries, for the login validation page. This job, when manually done, becomes hectic. But the new config element facilitates this generation automatically from the database specified.

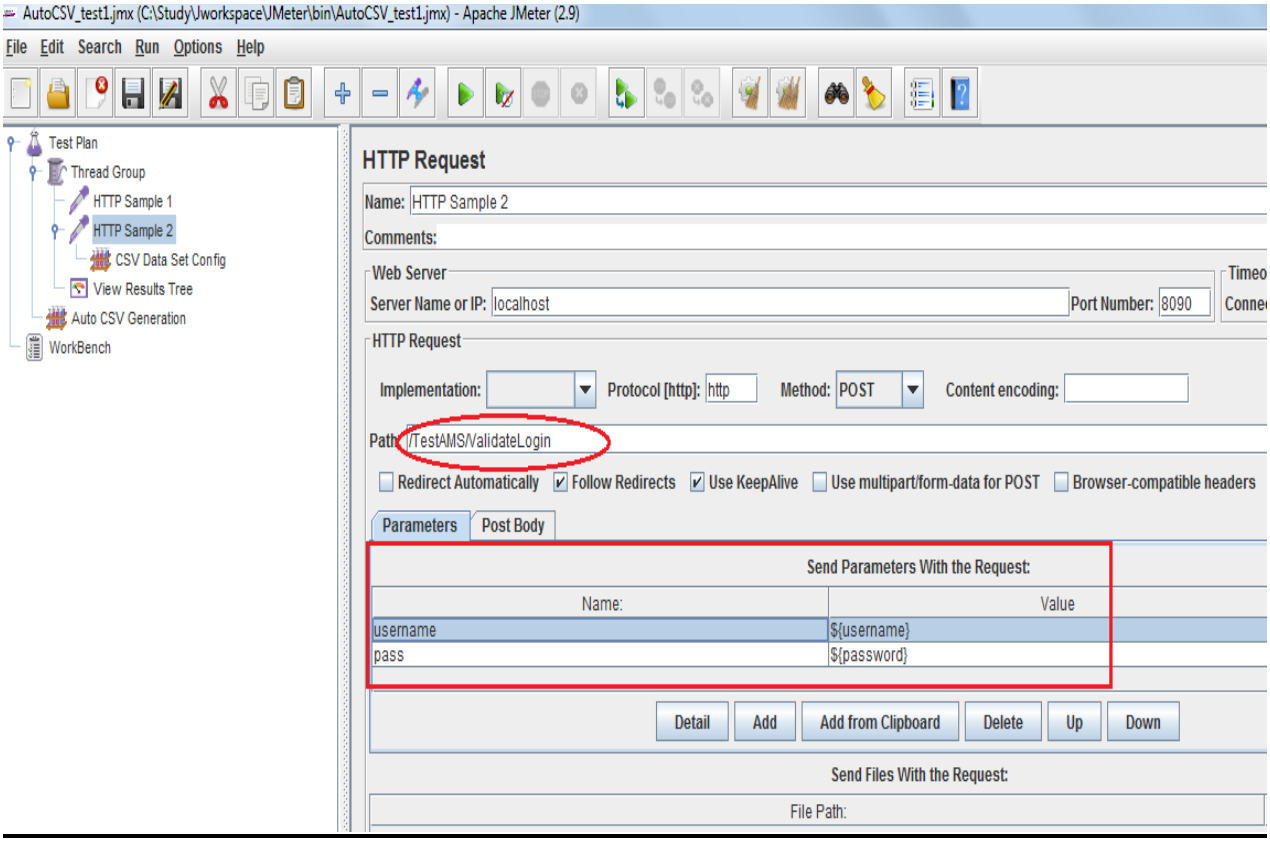
As we can see above, this csv file has already been created. Now this csv file needs to be clubbed with sampler, for its data to be used by sampler. The test plan below depicts it.

View4: Test plan with HTTP Samplers, to which "CSV data config" element is added as child:



The "CSV Data Set Config" Configuration Element is a configuration element already present in JMeter. In the above view, the comma separated variables are those variables whose values are picked up from the .csv file. These variables are also defined in the HTTP sampler which takes the "CSV Data Set Config" as child.

View5: The HTTP sampler, to which the .csv file is added as child:

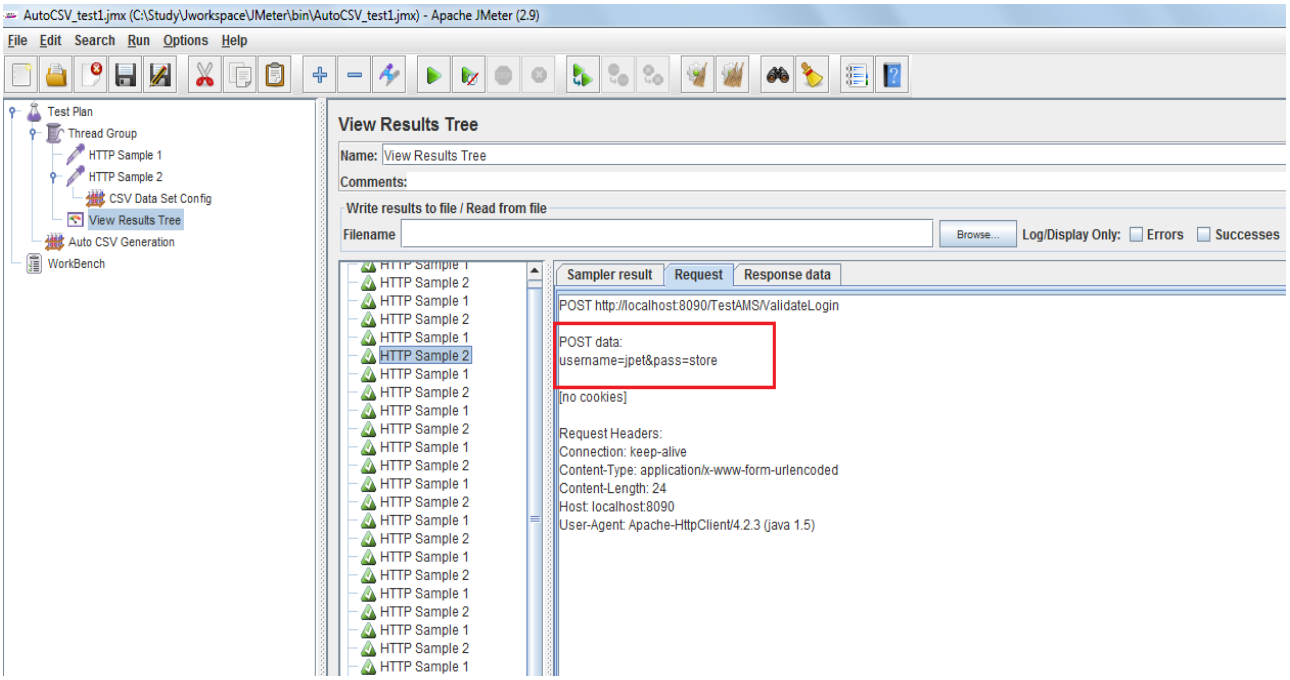


Here the parameters of request are set with variables mentioned in the "CSV Data Set Config".

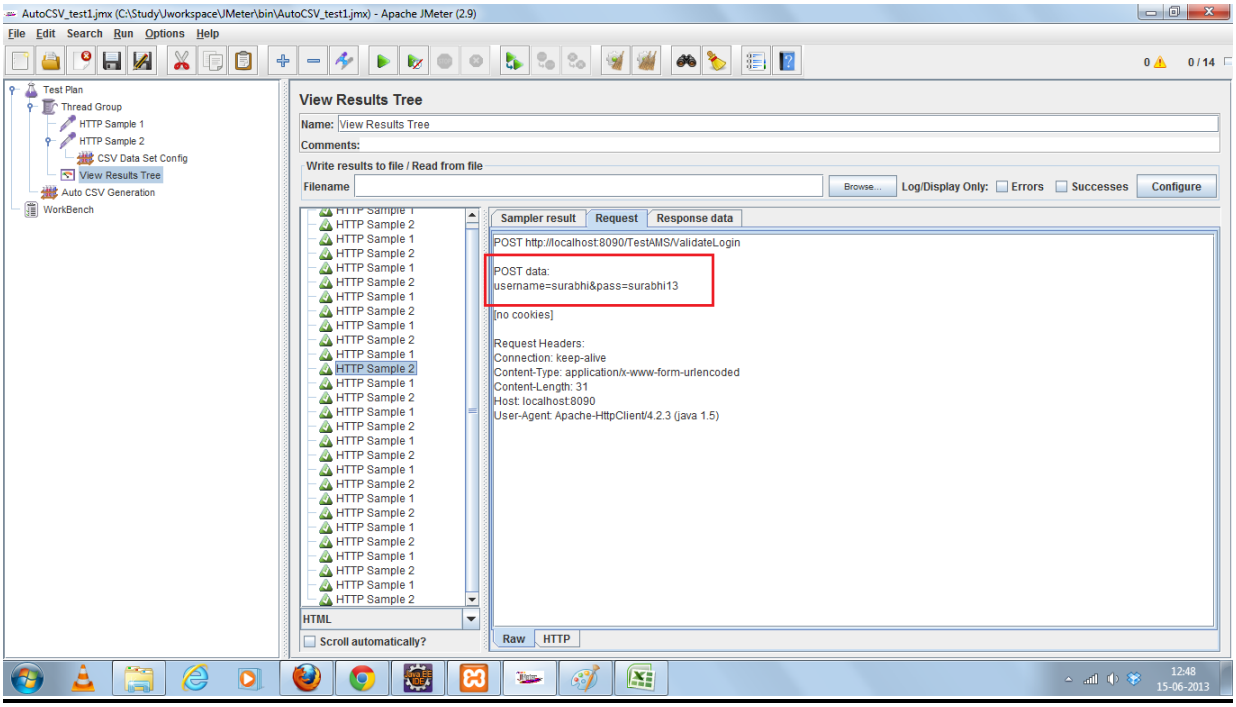
Then the test plan is played. The output of test is observed and it is seen that every request passes one set of values from .csv file as POST data, when ever that sample is used as user request.

<u>Observation:</u>

<u>View6: Observation 1:</u>



View7: Observation 2:

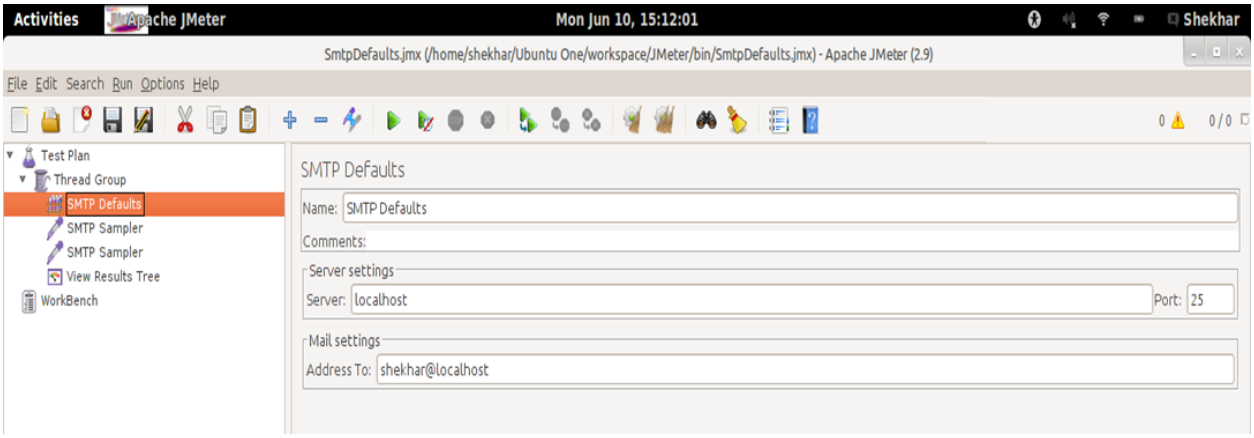


Conclusion:

Different requests take different set of data, successively from .csv file, and if the number of users is greater than the sets of data in the file, then the set of data are picked from the start of the file again. Thus the purpose of developing the new Auto CSV generation Config element was successful.

4.2 Test Plan: Demonstration of the working of "SMTP Defaults" Config Element

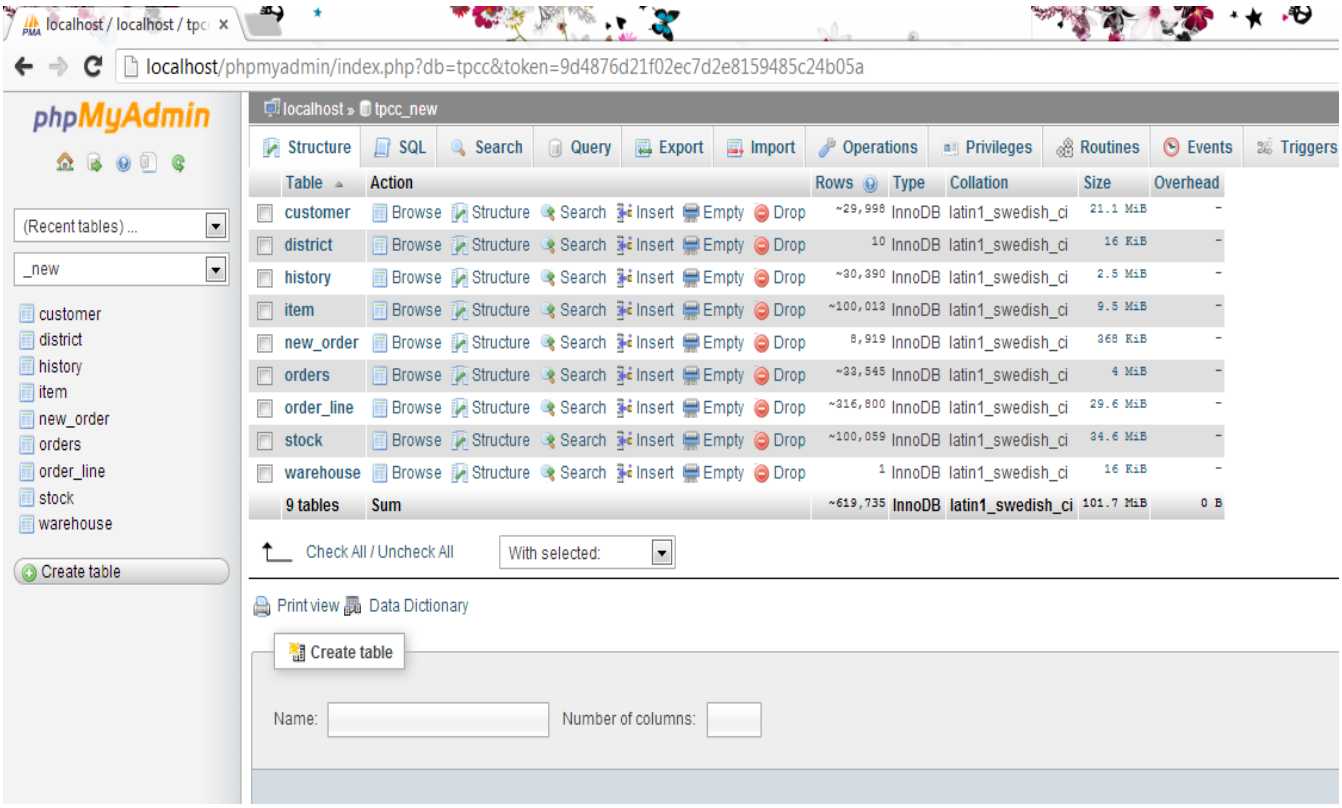
Aim: A test plan to demonstrate the working of SMTP defaults in Apache Jmeter.
System requirement: Apache Jmeter 2.9 and Postfix and Dovecot Mail server need to be installed on the system to be used for testing.
Procedure: The process to create the required test plan is described below
Step 1: A thread group is added to test plan from the Edit menu using Add → Thread group.
Step 2: Number of threads is set to 1 and Loop count is set to 1. See fig. 8.1
Step 3: An 'SMTP defaults' config element is added to thread group using Edit → Add → Config Element → SMTP Defaults
Step 4: Name of the element is left unchanged. In the server settings panel the server text box is set to localhost, and port number is set to 25. In the mail settings panel Address is set 'shekhar@localhost' which is an email address configured on localhost mail server. See fig. 8.2
Step 5: An SMTP sampler is added from Edit → Add → Samplers → SMTP Samplers. 'Address from' field can be set to any value. Here it is set to 'shekharsaurav@localserver.com'. The address format should be correct.
Step 6: In the message settings of the sampler. The subject of the message is set to 'Subject of the mail' and message text field is set to 'message of the mail'. See fig 8.3
Step 7: Another SMTP Sampler is added to the thread group and in the message setting panel, the value for subject is set to 'Subject for the mail 2' and message is set to 'message for the mail 2'. See fig 8.4
Step 8: A 'View Results Tree' listener is added to the thread group from Edit → Add → Listeners → View Results tree.



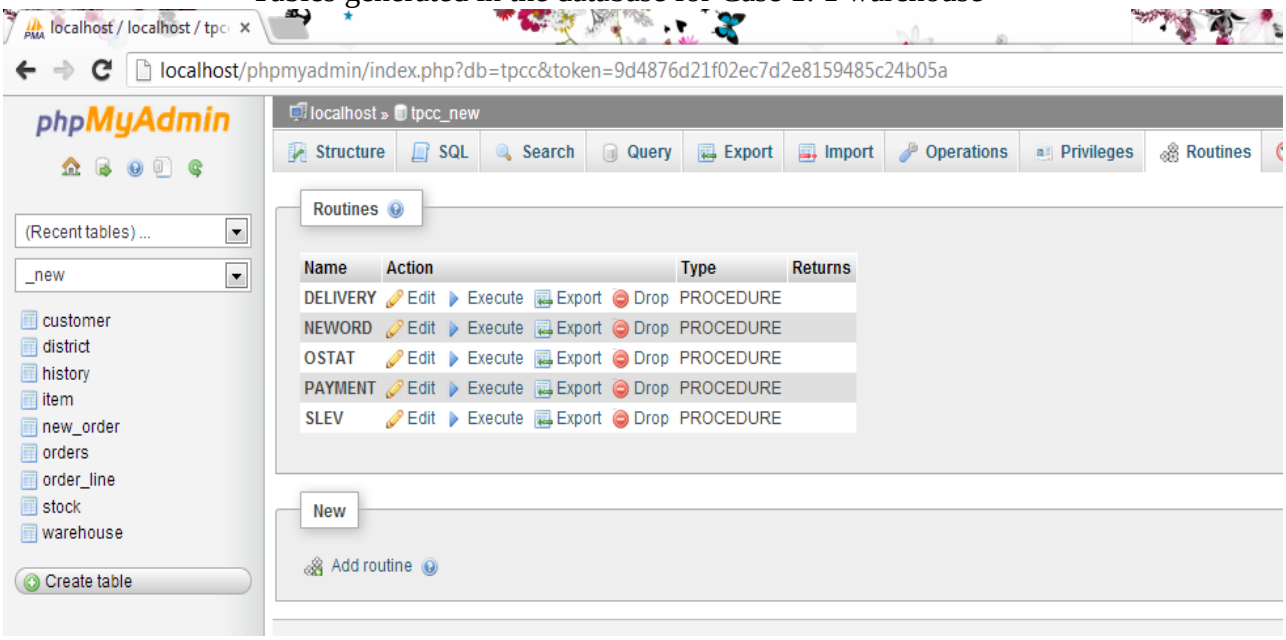
4.3 Automatic TPC-C testing in JMeter

Initially we tested the database by creating one warehouse. Here are the observations of the experiment.

As is clearly visible from the images shown below, the required tables and procedures with the required data get generated in the database. The general measure is that for 1 warehouse about 120MB of data is generated in the user's database, which gets multiplied depending on the number of databases.



Tables generated in the database for Case 1: 1 warehouse



List of routines generated in the database.

4.4 Test Plan: Demonstrating the working of "Bandwidth Throttling" in JMeter.

Aim: A test plan to demonstrate the working of Bandwidth Throttling in Apache Jmeter.
Procedure: The procedure to create a test plan to describe the working of Bandwidth throttling is described below

Step 1: Two thread groups were added in the test plan. Number of threads in each thread group was set to 5 with loop counts also as 5.
Step 2: In each thread group a HTTP request default config element were added. In one of the config element value of bandwidth is set to 1KBps and in the other config element of the other thread group the value of bandwidth was set to 1MBps and server of both of the config element was set to www.acmnitjsr.org
Step 3: In each of the thread group an http sampler was added
Step 4: In each of the thread group two types of visualizers were added, one was View results tree and aggregate report.
The complete test plan has been shown in the figure given below:

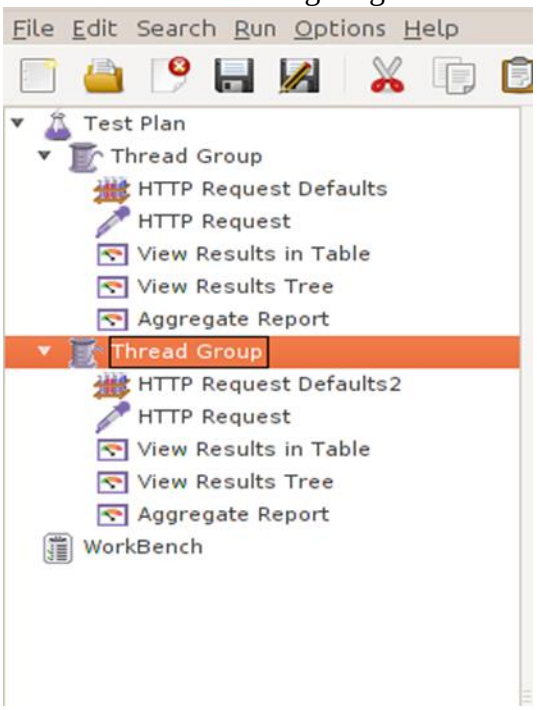


Fig: Test Plan hierarchy for Bandwidth Throttling demonstration

5 HUMAN INTERFACE DESIGN

5.1 Overview of User Interface

5.1.1 Auto CSV Generation

The Graphical User Interface of the Config Element – "Auto CSV Generation", takes the details of JDBC connection that has to be created with the Database from where data needs to be fetched to create the .csv file.

5.1.2 Filtered Results Listener Plugin

The GUI is quite simple; user can select the appropriate operator, specify the value and run the test.

5.1.3 SMTP Defaults

The graphical user interface for the SMTP defaults consist fields for the users to provide the defaults values to be set in the test plan. They are name of the component, comments, server name or address, port number or address, mail address to.

5.1.4 TPC-C Sampler

When the user clicks the ‘’create database’’ button depending on the number of warehouses generated the 9 tables mentioned in the database schema and the 5 procedures mentioned above get generated automatically and the tables also get filled with the required amount of data corresponding to the number of warehouses using random values as specified in the TPC-C specifications.

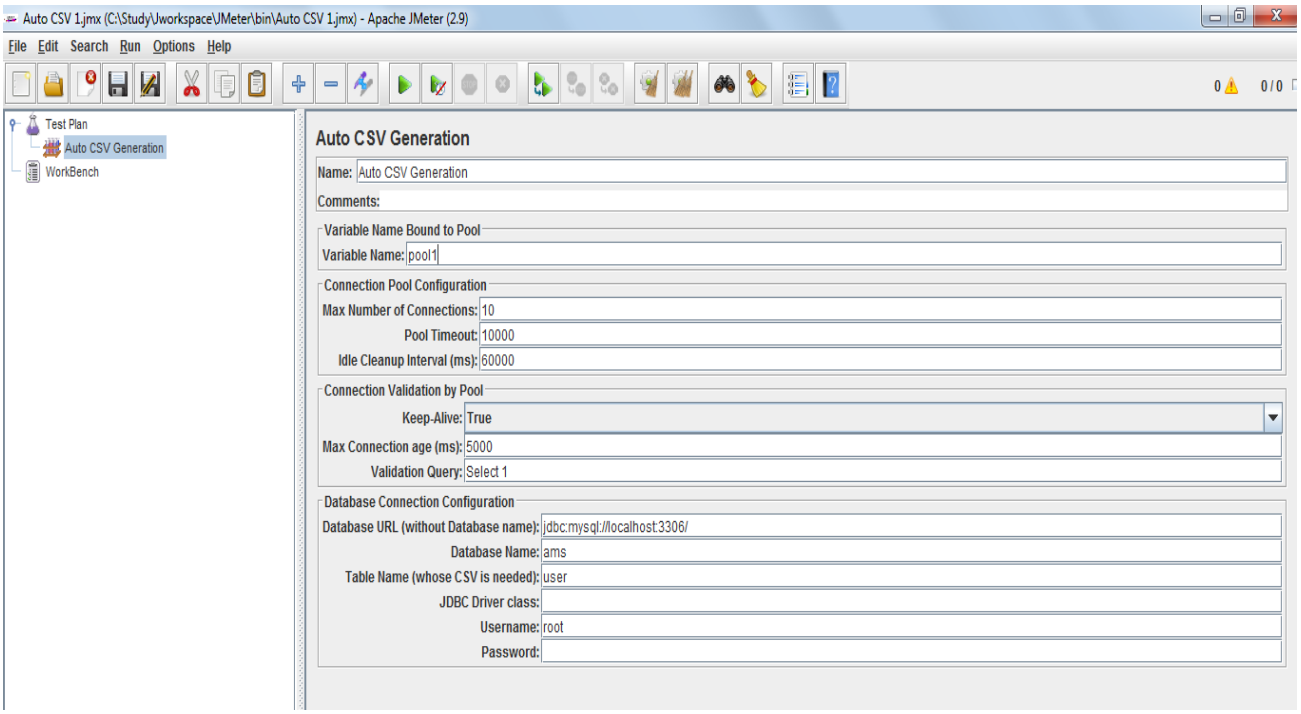
Similarly when the user clicks on the “start test” button the procedures which are generated in the database are called with parameters being random numbers and strings generated by the 15 functions added by us in the Jmeter source code.

5.1.5 Dynamic Bandwidth Throttling

Dynamic Bandwidth throttling element has been added with in the HTTP Request Defaults 'config element' of JMeter. It mainly consists of three components apart from the components which are already there in the config element.

5.2 Screen Images

5.2.1 Auto CSV Generation



- **Name:** The name that tester wishes to give to the element. It will be displayed on the "Test Plan Tree"
- **Variable name:** Name of that JMeter variable to which the connection pool will be bound to. It is the data source pool.
- **Max Number of Connections:** The maximum number of connections the pool will open at one time. By default it is set to 10.
- **Pool Timeout:** After this time period the pool blocks request for connection, until new connections are available. This is the maximum blocking time, until an exception is returned. It is in milliseconds. By default it is set to 10000ms (10sec).
- **Idle Cleanup Interval:** The pool removes extra idle connections at regular interval. This timing for interval is defined here. It is in milliseconds. By default it is set to 60000ms (60sec).
- **Keep Alive:** Whether the pool should validate connections. If no then the Connection Age and Validation Query are ignored.
- **Max Connection Age:** It is the maximum number of milliseconds an idle connection is kept, before discarding. It is in milliseconds. By default it is set to 5000ms
- **Validation Query:** A query used to validate if the connection is still alive. Relevant only if "Keep Alive" is true.
- **Database URL:** Full URL of the Database, including the JDBC protocol part, but excluding the database name only. The front slash "/" before the database name should be present.
- **Database Name:** The name of the database for which the .csv file of one of the table is to be created.
- **Table Name:** The name of the table of database for which the .csv file will be created.
- **JDBC Driver class:** Full package and class name of the JDBC Driver to be used. It must be included in the JMeter class path beforehand.
- **Username:** Username to use while connecting to database.
- **Password:** Password to use while connecting to database.

5.2.2 Filtered Results Listener Plugin

Figure 1: Old GUI without filtered results in "View Results in a Table"

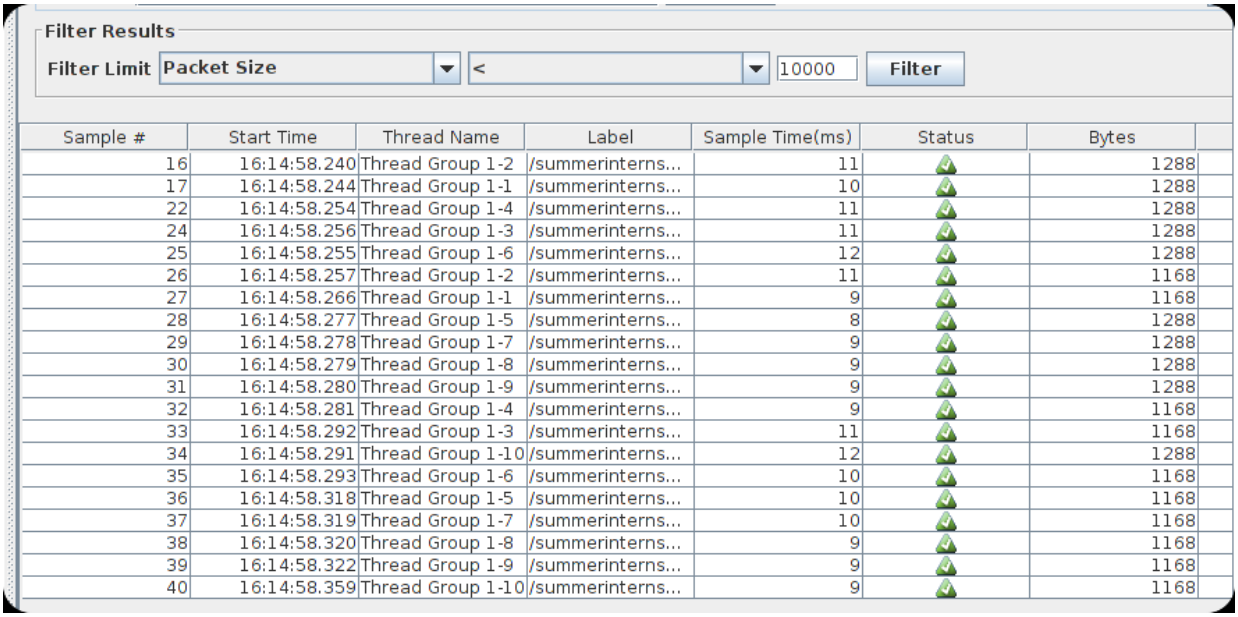
Filter Results

Filter Limit | 0 | Filter

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes
1	16:22:55.428	Thread Group 1-1	/summerinterns...	11		12585
2	16:22:55.433	Thread Group 1-3	/summerinterns...	8		12585
3	16:22:55.430	Thread Group 1-2	/summerinterns...	15		12585
4	16:22:55.438	Thread Group 1-4	/summerinterns...	8		12585
5	16:22:55.441	Thread Group 1-1	/summerinterns...	11		13878
6	16:22:55.446	Thread Group 1-7	/summerinterns...	7		12585
7	16:22:55.452	Thread Group 1-6	/summerinterns...	8		12585
8	16:22:55.453	Thread Group 1-8	/summerinterns...	8		12585
9	16:22:55.452	Thread Group 1-9	/summerinterns...	11		12585
10	16:22:55.446	Thread Group 1-5	/summerinterns...	19		12585
11	16:22:55.458	Thread Group 1-10	/summerinterns...	9		12585
12	16:22:55.455	Thread Group 1-3	/summerinterns...	13		13878
13	16:22:55.457	Thread Group 1-1	/summerinterns...	12		1288
14	16:22:55.456	Thread Group 1-4	/summerinterns...	13		13878
15	16:22:55.456	Thread Group 1-2	/summerinterns...	14		13878
16	16:22:55.457	Thread Group 1-7	/summerinterns...	14		13878
17	16:22:55.477	Thread Group 1-6	/summerinterns...	12		13878
18	16:22:55.478	Thread Group 1-8	/summerinterns...	13		13878
19	16:22:55.479	Thread Group 1-9	/summerinterns...	14		13878
20	16:22:55.480	Thread Group 1-5	/summerinterns...	14		13878
21	16:22:55.482	Thread Group 1-3	/summerinterns...	12		1288
22	16:22:55.483	Thread Group 1-1	/summerinterns...	12		1168
23	16:22:55.484	Thread Group 1-4	/summerinterns...	12		1288
24	16:22:55.484	Thread Group 1-2	/summerinterns...	12		1288
25	16:22:55.485	Thread Group 1-7	/summerinterns...	11		1288
26	16:22:55.481	Thread Group 1-10	/summerinterns...	16		13878
27	16:22:55.519	Thread Group 1-8	/summerinterns...	13		1288
28	16:22:55.518	Thread Group 1-6	/summerinterns...	14		1288
29	16:22:55.521	Thread Group 1-3	/summerinterns...	13		1168
30	16:22:55.520	Thread Group 1-5	/summerinterns...	15		1288
31	16:22:55.522	Thread Group 1-4	/summerinterns...	13		1168
32	16:22:55.519	Thread Group 1-9	/summerinterns...	16		1288
33	16:22:55.523	Thread Group 1-2	/summerinterns...	12		1168
34	16:22:55.523	Thread Group 1-7	/summerinterns...	12		1168
35	16:22:55.524	Thread Group 1-10	/summerinterns...	12		1288
36	16:22:55.584	Thread Group 1-6	/summerinterns...	9		1168
37	16:22:55.583	Thread Group 1-8	/summerinterns...	11		1168
38	16:22:55.585	Thread Group 1-5	/summerinterns...	10		1168
39	16:22:55.587	Thread Group 1-9	/summerinterns...	9		1168
40	16:22:55.589	Thread Group 1-10	/summerinterns...	9		1168

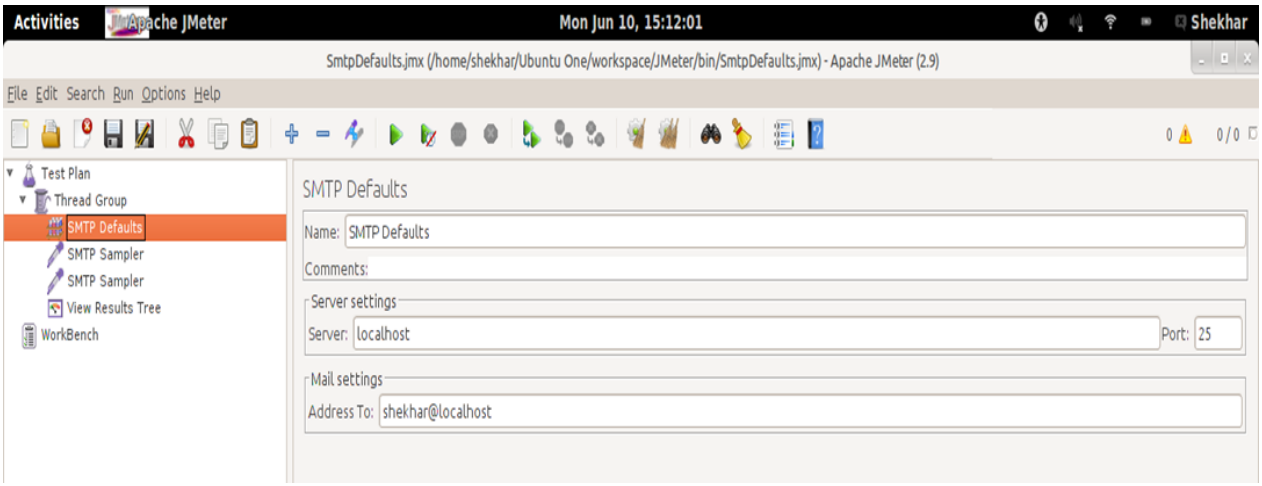
Scroll automatically? Child samples? No of Samples 40 Latest Sample 9 Average 11 Devia

Figure 2: New GUI with filter 'Limit' in "View Results in a Table"



Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes
16	16:14:58.240	Thread Group 1-2	/summerinterns...	11		1288
17	16:14:58.244	Thread Group 1-1	/summerinterns...	10		1288
22	16:14:58.254	Thread Group 1-4	/summerinterns...	11		1288
24	16:14:58.256	Thread Group 1-3	/summerinterns...	11		1288
25	16:14:58.255	Thread Group 1-6	/summerinterns...	12		1288
26	16:14:58.257	Thread Group 1-2	/summerinterns...	11		1168
27	16:14:58.266	Thread Group 1-1	/summerinterns...	9		1168
28	16:14:58.277	Thread Group 1-5	/summerinterns...	8		1288
29	16:14:58.278	Thread Group 1-7	/summerinterns...	9		1288
30	16:14:58.279	Thread Group 1-8	/summerinterns...	9		1288
31	16:14:58.280	Thread Group 1-9	/summerinterns...	9		1288
32	16:14:58.281	Thread Group 1-4	/summerinterns...	9		1168
33	16:14:58.292	Thread Group 1-3	/summerinterns...	11		1168
34	16:14:58.291	Thread Group 1-10	/summerinterns...	12		1288
35	16:14:58.293	Thread Group 1-6	/summerinterns...	10		1168
36	16:14:58.318	Thread Group 1-5	/summerinterns...	10		1168
37	16:14:58.319	Thread Group 1-7	/summerinterns...	10		1168
38	16:14:58.320	Thread Group 1-8	/summerinterns...	9		1168
39	16:14:58.322	Thread Group 1-9	/summerinterns...	9		1168
40	16:14:58.359	Thread Group 1-10	/summerinterns...	9		1168

5.2.3 SMTP Defaults



Fields in SMTP defaults GUI:

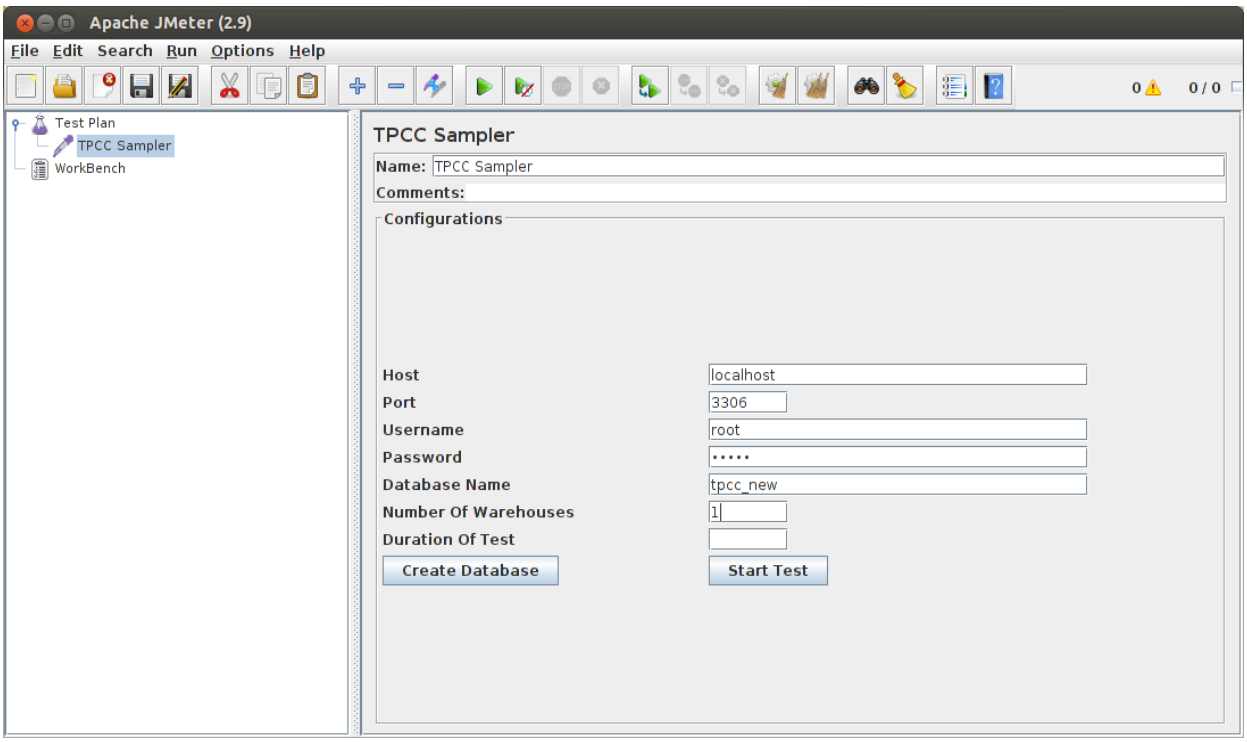
1. Name: Name of the config element in the test plan. This name is used to identify the element in the tree view of the test plan
2. Comments: This text field is provided to give some related comment or description of the element used in the test plan. It may describe the use of the element in the test plan.
3. Server Settings Panel:
    a. Server: This text field is used specify the IP address or the domain name of the mail server where all the mails under this config element is to be sent. This value is used by all the smtp samplers with in this config element in the hierarchy unless a sampler has specifically specified the value for server.

b. Port: This text field is used to specify the port address on which the mails are to be sent.

Once specified, all the samplers in the test plan will use the same port address unless a sampler has specifically specified the value for port.

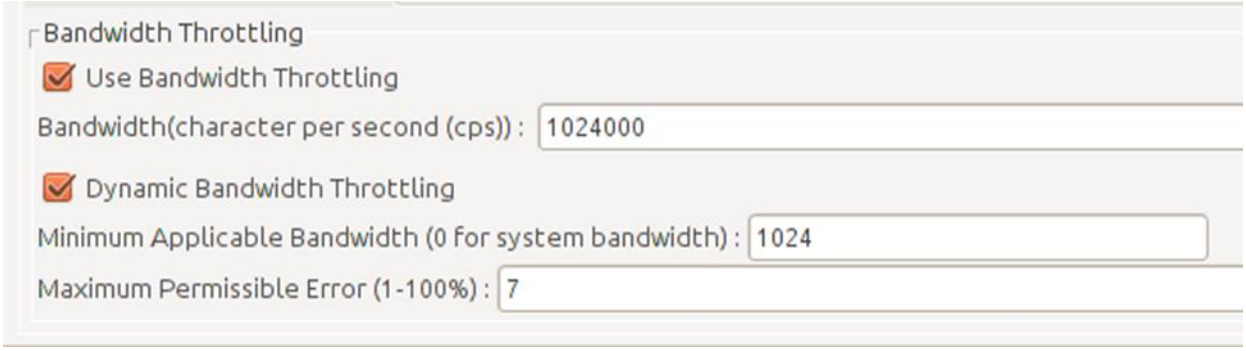
4. Mail Settings Panel
   a. Address To: This text field is used to specify the mail address where all the mails from the samplers in the test plan hierarchy will be delivered. Once specified, this value is used by all the samplers in the hierarchy unless a sampler has specifically specified the value for address to.

5.2.4 TPC-C Sampler



The above figure shows the GUI for the sampler being generated. It takes as parameters the host name , port number, username , password , name of the database to be created and the number of warehouses to be generated and the total duration of the test.

5.2.5 Dynamic Bandwidth Throttling



- **'Dynamic Bandwidth Throttling' Check Box**: This check box is used to set the use of dynamic bandwidth with Jmeter. Once selected, this enables the use of dyanmic bandwidth variations. Enabling this component also enable the text fields in the GUI.
- **'Minimum Applicable Bandwidth'**: This sets the minimum level for the bandwidth upto which the bandwidth can be reduced during runtime. Default value is 0.
- **Maximum Permissible Error**: This text box is used to specify the threshold value which will be used to vary the bandwidth. If the error crosses this value, then the bandwidth starts decreasing until the error again gets below this value or reaches the minimum applicable bandwidth value specified in the previous text field. Default is 100%.

5.3 Screen Objects and Actions

6 REQUIREMENTS

The Operating System used by us is Ubuntu 12.04, The dependencies for the building and execution of the source code are as follows

- Apache JMeter 2.9 requires JDK5 or above
- Ant version 1.8 or above is required to build the project from the provided build.xml file

Before building the project some libraries need to be added and updated using the command: ant download_jars.