
## Assignment 9

# Load Test and Performance Measurements

---

**Due Date: April 6, 2020**

**Yasaman Amannejad, 2020**

TABLE OF CONTENTS

# 1 INTRODUCTION

Many systems ranging from e-commerce websites to telecommunication infrastructures must support concurrent access by hundreds or thousands of users. Many of the problems in these systems are related to systems not scaling with the increase of their workloads. To assure the quality of these systems, load testing is a required testing procedure in addition to conventional functional testing procedures, such as unit, integration, and system testing. Load testing, in general, refers to the practice of assessing a system's behavior under various levels of load on the system. A load is typically based on a usage profile, which describes the number of users and the type of requests they submit to the system. A usage profile, describes the types of executed scenarios and the rate of these scenarios. Load testing requires one or more load generators which mimic clients sending thousands or millions of concurrent requests to the application under test. During the course of a load test, the application is monitored and performance data along with execution logs are recorded. Performance data store resource usage information such as CPU utilization, memory, disk I/O and network traffic, and also application level metrics such as response time and throughput.

## 1.1 OBJECTIVES

This assignment is an introduction to some of the concepts inherent to software performance testing. Specifically, students should gain an understanding of some fundamentals involved in performance testing. After taking this assignment, students will learn:

- how to design their performance tests.
- how to generate workload for a Web application.
- how to record performance testing scenarios.
- how to collect and analyze their performance testing results.

## 1.2 GROUP-WORK SPECIFICATION

This assignment is a group activity. Students can complete this assignment in groups of 2-3 students.

## 1.3 SOFTWARE UNDER TEST

Please use version 1.1 of JPetStore (from assignment 1) as your application under test. You are going to put load on this application and record the performance metrics. Each team should run JPetStore on their Cloud platform.

## 1.4 ASSIGNMENT PROCESS

In this assignment, you are going to create load test scripts for two type of load profiles given in task 1 and task 2 (Section 3). Then, you will run your tests on JPetStore and collect performance metrics.

Section 2 of this document is written to help you learn working with JMeter. If you know how to work with JMeter, you can skip this section and move directly to Section 3 – Instructions, and complete your task.

## 1.5 DELIVERABLES

For each task, submit your script (jmx file), and also include all your results (numbers, graphs, screenshots) in a Word file. In your Word document, also clarify the type of cloud machine you used for your load tests, e.g., how much CPU, or memory was assigned to your virtual machine.

# 2 WORKING WITH JMETER

## 2.1 INTORDUCTION TO APACHE JMETER

**Apache JMeter** - Apache JMeter is an Apache project that can be used as a load testing tool for analyzing and measuring the performance of a variety of services, with a focus on Web applications. JMeter supports variable parameterization, assertions (response validation), per-thread cookies, configuration variables and a variety of reports.

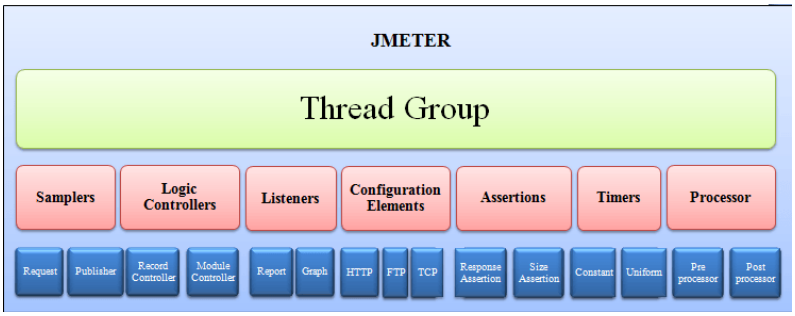
JMeter is a java-based desktop application. You can download it from https://jmeter.apache.org/download_jmeter.cgi and run the JMeter jar file under the bin folder. For more information and documentation, please refer to JMeter Website:

- Website: https://jmeter.apache.org/
- Documentation: https://jmeter.apache.org/usermanual/get-started.html

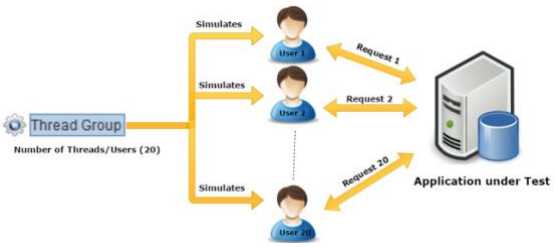
The rest of this section describes some of the important features in JMeter.

## 2.2 JMETER ELEMENTS

Different components of JMeter are called Elements. Each Element is designed for a specific purpose. The figure below shows some common elements in JMeter.




**Thread Group** - Thread Groups is a collection of "threads". Each thread simulates one real user working with the system. For example, if you set the number of threads as 20, JMeter will create and simulate 20 virtual users (vUser). Each user can interact with the server by submitting their requests.

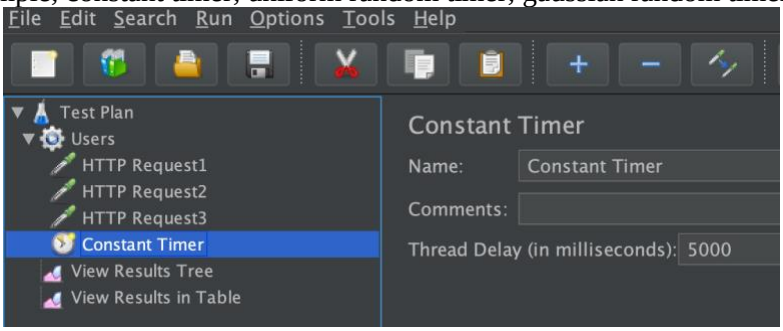



**Samplers** – Samplers in JMeter allows JMeter to send different types of requests to a server. Samplers are the requests that JMeter sends (on behalf of vUsers) to the Web server under test. The sample results have various attributes (success/fail, elapsed time, data size etc.). There are different types of samplers in JMeter. To send a HTTP request to a Web server, you can use HTTP request sampler.

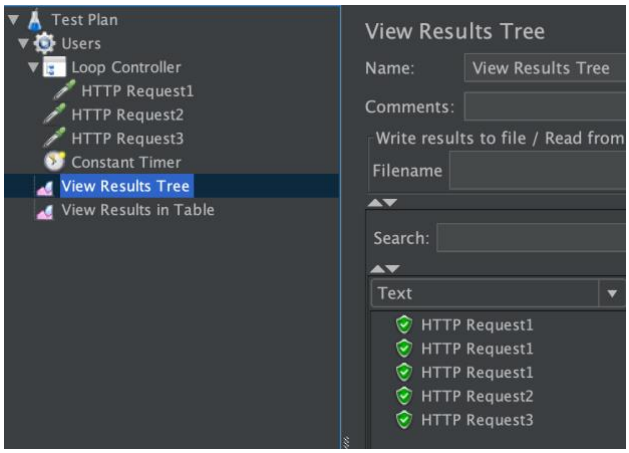
**Listeners** – Listeners show the results of the test execution. They can collect and show results of different performance metrics in different formats such as a tree, table, graph, and log file.

**Configuration** – are used for setting the default variables that can be used by samplers, e.g., datasets, URL, etc.

**Timer** – can help you to apply wait time between each request that a vUser submits. In fact, the timer simulates the user "think time" between the consecutive pages that will visit. There are various timers available, for example, constant timer, uniform random timer, gaussian random timer, etc.



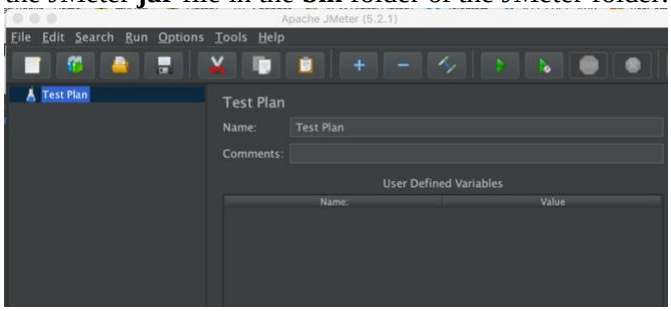

**Logical controllers** – let you define the order of requests in a Thread. It lets you control "when" to send a user request to a web server. For example, you can use Random Controllers to send HTTP requests to the server randomly, or you can add a loop controller to repeat a request multiple times. In the following figure, a loop controller is added around request 1. Therefore, you can see in the output that request 1 is submitted 3 times.




## 2.3 CREATING YOUR FIRST JMETER TEST

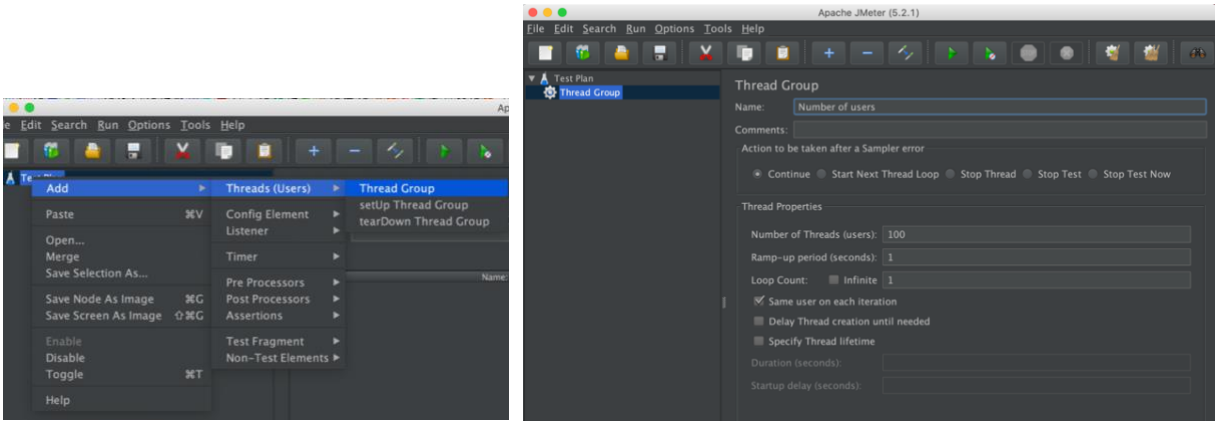
In this section, we will create our first JMeter test to do a performance analysis of JPetStore login page.

1. Start JMeter using the JMeter **jar** file in the **bin** folder of the JMeter folder.

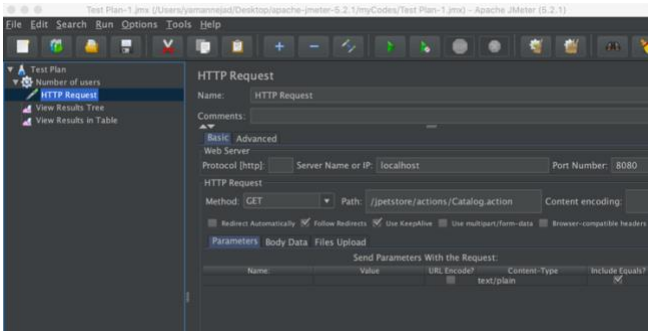



2. Create a **test plan** if you do not have one in the left side of your JMeter window.

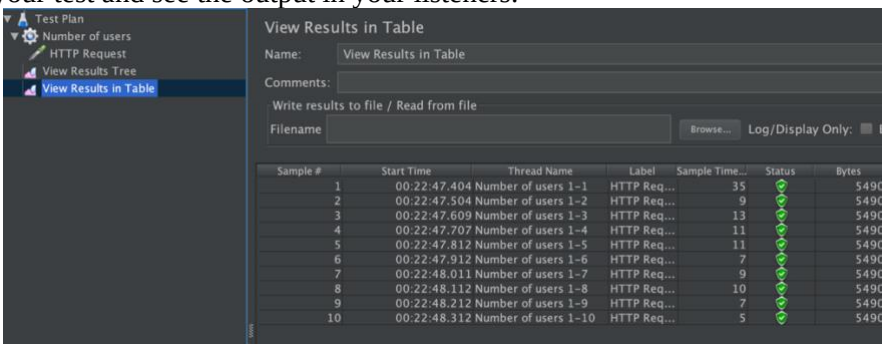
3. Create a **thread group** by right-click on your test plan and adding a thread group with X number of users.




4. After defining users, now we should define what type of requests should we send by vUsers. Right-click on your thread group and add a HTTP request **sampler**. Complete the details of your HTTP request.




5. Now, create **listeners** to collect the outputs. You can create more than one type of reports, e.g., a tree and a table format report.
6. Save your test plan as a jmx file.
7. Run your test and see the output in your listeners.



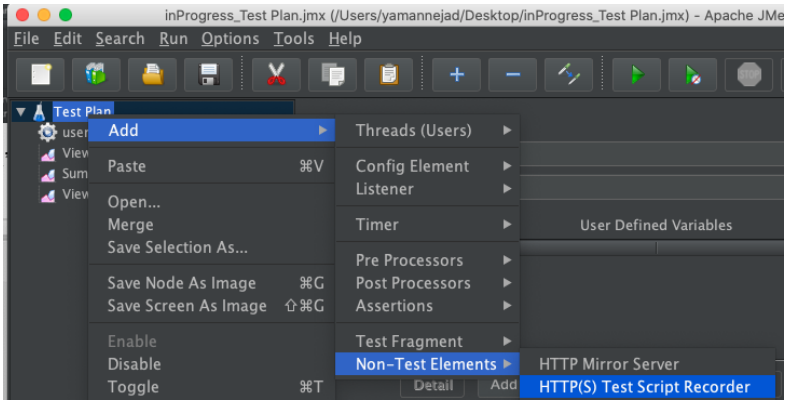

8. You can change the thread group configurations to see the effect of the number of submitted requests on the system and the collected results. Before running with a new configuration, make sure to delete the previously collected outputs.

## 2.4 RECORDING WITH JMETER

In this section, we will record the operations that vUsers will do in the system. You can record a script directly in JMeter[1]. The steps that you need to follow are:

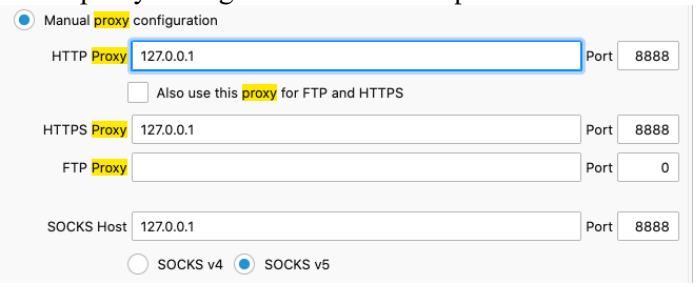
1. Right click on your test plan and add Non-Test element - http test recorder

[1] There are also tools available that allow you to generate jmx scripts and import into JMeter. One such tool is Blazemeter. Here is a video about using JMeter and Blazemeter together: Watch this video



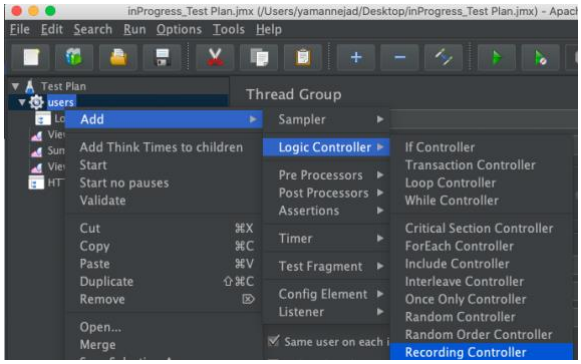

2. Set proxy on Browser

Scripts will be recorded based on your interaction with the browser. To capture this, JMeter needs to work as a proxy. It will sit in between your browser and the Webserver and will record your actions on the browser. You can set proxy for your Google Chrome using: Chrome → Setting → Advanced → System's proxy setting. There is similar option in Firefox as well.



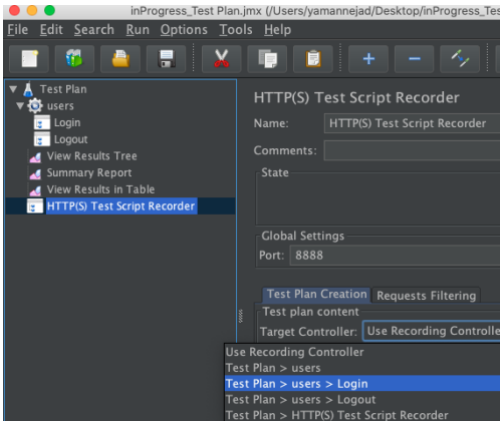

3. Create recording controllers

Before recording your script, it is good to break it down into smaller modules using Logic Controllers → Recording Controllers. This way you can have modular scripts for your performance tests.

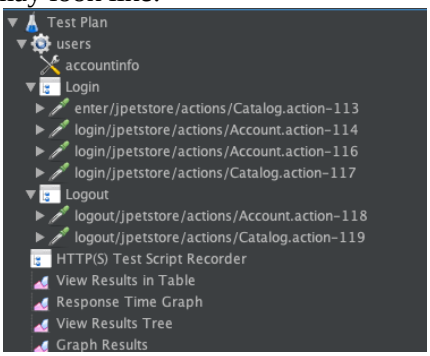



4. Start recording in different recording controller

To select where your interactions should be saved, you can select the target controller in your HTTP Test Script Recorder. Select your target, work with your browser and this will record your script. Stop recording when your interactions are completed.




Here is how your final test plan may look like:



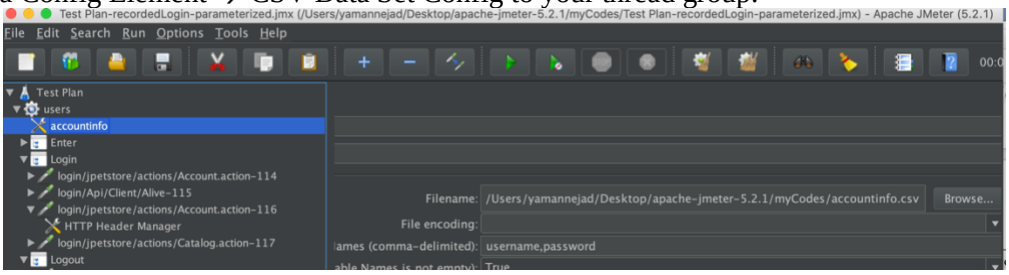

## 2.5 PARAMETERIZE YOUR JMETER SCRIPT

After recording your script, you can parameterize it to run it with different users and input data.

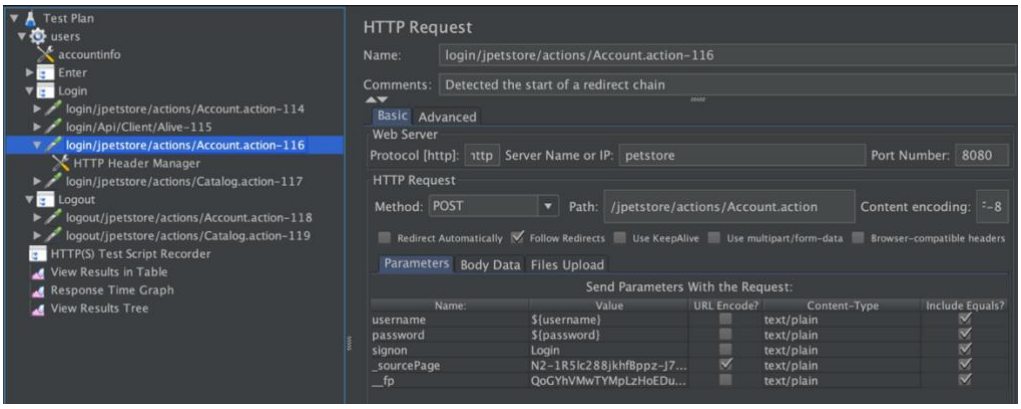
1. To do this, first create a CSV file, e.g.,

| | A | B | C | D |
|---|---|---|---|---|
| | **username** | **password** | | |
| | quality1 | 12345678 | | |
| | j2ee | j2ee | | |
| | quality2 | 12345678 | | |

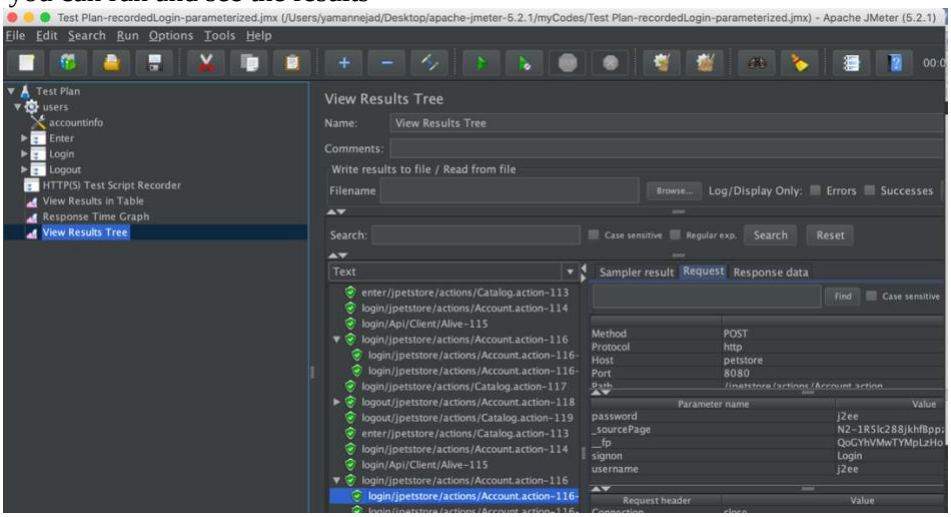
2. Add a Config Element → CSV Data Set Config to your thread group.




3. Parameterize your script – For each user input, assign a column from your CSV file from where the data should be inserted into your scripts. To do this, you can replace the value of a parameter with ${columnName}, where "columnName" is the name of the column from your CSV file that you want to use for this input variable.




4. Finally, you can run and see the results




## 2.6 CONTROLLERS IN JMETER

You can control the order and number of times each recorded module runs. This is important for defining different user scenarios and system usage patterns. Some of the important controllers are:

**Simple Controller** - is just a container for user request. You can group scripts of different features under this controller and all features will run in a regular format and order.

**Recording Controller** – This is the type of controller that we used for recording our scripts.

**Random Controller** – Allows you to run a set of scripts randomly. It means that in each run of the test only one of the scripts under the random controller will run.

**Interleave Controller** – Allows you to run one script from a group of scripts. The scripts will be selected to run in a round robin order, and not randomly.

**Random Order Controller** – Allows you to run a set of scripts in a random order. All scripts will run with this controller, but their order of execution will be random.

**Throughput Controller** – Allows you to distribute users among different scripts. **This can be useful for defining the usage pattern of your vUsers**.

**Module Controller** – Sometimes you may want to use one of your scripts multiple times. Module controller will allow you to do this without duplicating it.

# 3 INSTRUCTION

### Task 1)

1. Create a testing scenario that can be used to create load on JPetStore, with the following load profile:
   - (Type 1 users) 40% of JPetStore users will only browse the system without login into the system.
   - (Type 2 users) 40% of JPetStore users will browse the system after login into the system.
   - (Type 3 users) 20% of JPetStore users will make a purchase after they login to the system.

2. Parameterize your tests to read the username and password, and the products to browse and purchase from a CSV file.

3. Create 10 users in the system (You can do this manually in the Register new user page).

4. Run your test with your 10 users, and run your test for a couple of minutes (Loop count = Infinite) and collect the performance data:

   - What is the average, 90%, and 95% response time for different HTTP requests?
   - What is the throughput of the system under test?
   - What is the CPU usage of the system during this performance test?

### Task 2)

5. If all JPetStore users were of type 1 users (only browsing), how would be the response time (average, 95% percentile response time) of the system change when 100 concurrent users are in the system vs when 1500 users are in the system. Repeat each test (100 user, 1500 user), 3 times and report your results in each round.
6. Are you getting consistent results?
7. Increase the number of users from 100 to 1500 in steps of 100, and run your tests to collect the average response times. Draw a plot that shows how your response time is changing by the increase of the number of users in the system.

# 4 SUBMISSION

## 4.1 Deliverables

- For each task, submit your script (jmx file), and also include all your results (numbers, graphs, screenshots) in a word file.
- In your word document, also write down the information of your JPetStore cloud instance, e.g., how many CPU, and how much memory were assigned to your virtual machine.

## 4.2 Grading

| | |
|---|---|
| Task 1 – correctness of the script | 25% |
| Task 1 – parameterized script is used correctly | 10% |
| Task 1 – performance data extracted and analyzed properly | 30% |
| Task 2 - correctness of the script | 10% |
| Task 2 – parameterized script is used correctly | 5% |
| Task 2 – performance data extracted and analyzed properly | 20% |