**Performance Testing tool**

**Why JMeter?**

* Apache JMeter is a versatile open-source automation tool primarily used for performance testing, load testing, and functional testing of web applications, APIs, and other services.
* Java-based, which can be resource-intensive, especially with large user loads.
* Should only Navigate through UI, to generate test scripts and are cumbersome to maintain or track changes in Version Control Systems.
* Supports most protocols.
* Provide IP Spoofing — The JMeter host must have multiple IP addresses (i.e. IP aliases, network interfaces, devices)
* Inbuilt support to control traffic to different endpoints in each Thread group using Logic Controllers

**Here's what JMeter covers in terms of Automation:**

**Performance Testing**: JMeter allows testers to simulate heavy loads on web servers, databases, and other services to measure their performance under various conditions. It can generate a large number of virtual users to mimic real-world scenarios and assess how the system performs under load.

**Load Testing:** Load testing with JMeter involves applying a load to the system and measuring its response. Testers can simulate different load levels, ramp-up patterns, and concurrency scenarios to evaluate the system's capacity, scalability, and reliability.

**Functional Testing:** JMeter can be used for functional testing to verify that web applications and APIs behave as expected. Testers can create test scripts to send requests, validate responses, and verify functionality such as login, registration, search, and checkout processes.

**Protocol Support:** JMeter supports a wide range of protocols including HTTP, HTTPS, FTP, JDBC, LDAP, SOAP, REST, TCP, and more. This allows testers to automate testing of various types of applications, services, and network protocols.

**Scripting and Test Automation:** JMeter provides scripting capabilities that allow testers to create and customize test scenarios using its built-in scripting language (Beanshell) or by recording user interactions with the application. Test scripts can be automated and executed repeatedly to ensure consistent testing.

**Parameterization and Data-Driven Testing:** JMeter supports parameterization and data-driven testing, allowing testers to vary test data and input parameters across multiple iterations or scenarios. This helps in testing different use cases and identifying potential issues related to data handling and processing.

**Assertions and Validation:** JMeter offers various assertion types to validate the correctness of responses and behavior during testing. Testers can define assertions to verify HTTP status codes, response times, content, headers, and other aspects of the application or service under test.

**Reporting and Analysis:** JMeter provides built-in listeners and reporting tools to monitor test execution in real-time and generate comprehensive test reports. Testers can analyze test results, identify performance bottlenecks, and make data-driven decisions to optimize system performance.

**Integration with CI/CD Pipelines**: JMeter can be integrated into Continuous Integration/Continuous Deployment (CI/CD) pipelines to automate the execution of performance tests as part of the software development lifecycle. This ensures that performance testing is conducted consistently with each code change.

**Overall, JMeter covers a wide range of testing scenarios and offers powerful automation capabilities for performance testing, load testing, and functional testing of web applications, APIs, and services. Its flexibility, scalability, and ease of use make it a popular choice for organizations looking to automate their testing efforts and ensure the quality and performance of their software products.**

**When to use/consider JMeter?**

**Performance testing is a crucial aspect of ensuring that software applications meet the demands of real-world usage. Among the various tools available for performance testing, Apache JMeter stands out for its versatility and capabilities. One of the essential features in JMeter is the ability to record user interactions using a proxy server. However, for accurate performance testing, incorporating think time is paramount. In this article, we will explore the significance of think time and how to efficiently utilize it during recording in JMeter.**

**Response Time:** Response time is the time taken for the server to respond to a request sent by the client. It includes the time spent on processing the request on the server, network latency, and time taken to receive the response back to the client.

**Latency:** Latency is the time taken for a request to travel from the client to the server and back. It represents the network delay between the client and server and is often measured separately from the server's processing time.

**Throughput:** Throughput is the number of requests processed by the server per unit of time. It measures the system's capacity to handle a certain volume of traffic and is typically expressed in requests per second (RPS) or transactions per second (TPS).

**Concurrency:** Concurrency refers to the number of simultaneous users or connections accessing the server at a given time. JMeter can simulate concurrent users to assess how the system performs under different levels of concurrency.

**Error Rate:** Error rate measures the percentage of requests that result in errors or failures. It indicates the system's reliability and stability under load and helps identify issues such as server errors, timeouts, or connection failures.

**CPU Utilization**: JMeter can monitor the CPU utilization of the server during performance testing to assess how efficiently the server's CPU resources are being utilized. High CPU utilization may indicate performance bottlenecks or resource constraints.

**Memory Utilization:** Memory utilization measures the amount of memory consumed by the server process during performance testing. JMeter can monitor memory usage metrics to identify memory leaks, excessive memory consumption, or inefficient memory management.

**Network Bandwidth**: JMeter can measure the network bandwidth consumed by the client and server during performance testing. It helps identify network-related issues such as bandwidth constraints, network congestion, or packet loss.

**Database Performance:** JMeter can be used to measure the performance of database queries and transactions. It can simulate database interactions and measure parameters such as query execution time, throughput, and error rate.

**Page Load Time:** For web applications, JMeter can measure the page load time, which includes the time taken to download and render the web page in the client's browser. It helps assess the user experience and performance of web applications.

**These are some of the key performance parameters that can be measured using JMeter. By analyzing these parameters, testers can identify performance bottlenecks, optimize system performance, and ensure the reliability and scalability of their applications and services.**

**Understanding Think Time**

Think time is the pause or delay between consecutive user actions in an application. It simulates how real users interact with an application, adding a level of realism to performance testing. When users interact with an application, they don’t instantly click on the next action; instead, they take a moment to process the information or perform an action.

In performance testing, replicating this behavior is crucial to generate accurate load scenarios. By incorporating think time during recording, we can emulate the natural rhythm of user interactions, ultimately leading to more realistic performance testing results.

**The Importance of Think Time in Performance Testing**

**Realism in Load Simulation:** Think time replicates the actual behavior of users, ensuring that load testing closely resembles real-world scenarios. This helps in identifying potential performance issues that might occur in production.

**Server Load Estimation:** Accurate think time allows for a more precise estimation of the server load. It helps in simulating the load that the application will experience in the production environment, aiding in capacity planning.

**Stress Testing Accuracy:** During stress testing, simulating a sudden influx of users is more accurate with think time. It allows for a better understanding of how the application behaves under stressful conditions.

**Simple Architectural diagram for JMeter framework**

A screenshot of a computer program

Description automatically generated

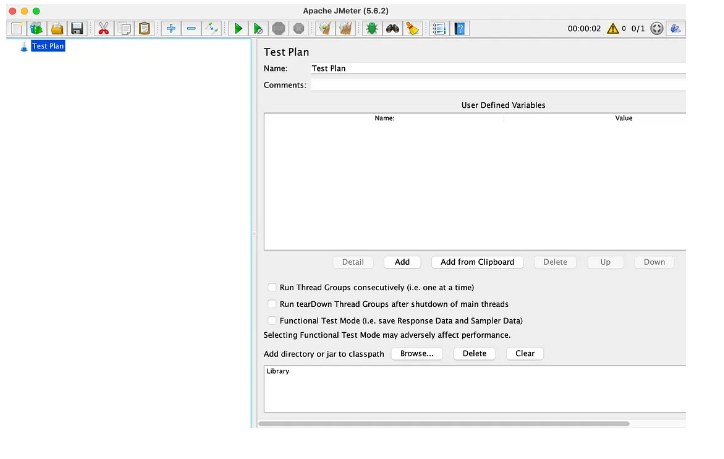
**How to Use JMeter? With Example.**

**Incorporating Think Time in JMeter Recordings**

Let’s go through the steps to efficiently incorporate think time during the recording process in JMeter:

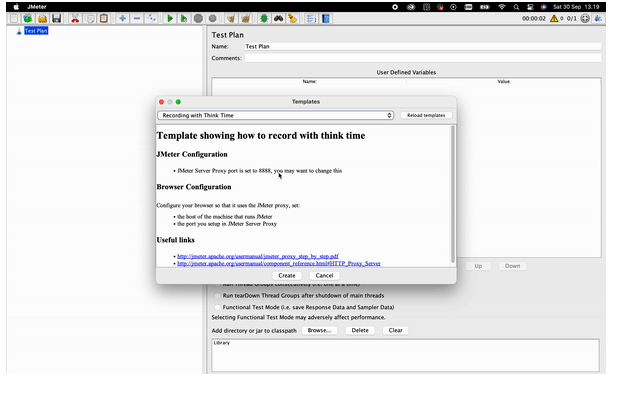
**1.Create a New Test Plan in JMeter:**

. Begin by initiating a new test plan in JMeter.



**2. Access the Template for Recording With Think Time:**

Navigate to the ‘File’ menu, then select ‘Templates,’ and opt for ‘Recording With Think Time.’ Click ‘Create’ to proceed.

**3. Rename the Recording Controller:**

Expand the ‘Thread Group’ and select the ‘Recording Controller.’ Rename it accordingly by clicking on the controller.

A screenshot of a computer

Description automatically generated

**4. Configure the HTTP(S) Test Script Recorder:**

Access the ‘HTTP(S) Test Script Recorder.’ Adjust the ‘Target Controller’ within the ‘Test Plan Content’ section to point to the renamed ‘Login Test’ Recording Controller.

A screenshot of a computer

Description automatically generated

**5. Apply URL Filters to Requests:**

Optionally, include or exclude specific URL patterns for your requests by navigating to the ‘Request Filtering’ section, located beside the ‘Test Plan Creation’ section.

A screenshot of a computer

Description automatically generated

**6. Return to Test Plan Creation:**

Go back to the ‘Test Plan Creation’ section to continue configuring your test plan.

**Recording HTTP(S) Request on Browser**

When it comes to recording HTTP(S) requests for performance testing, leveraging a web browser to mimic real user interactions is a popular approach. In this guide, we’ll demonstrate how to use Firefox for this purpose, given its ease of setup compared to Chrome.

**1.Launch the browser and access the settings.**A screenshot of a computer

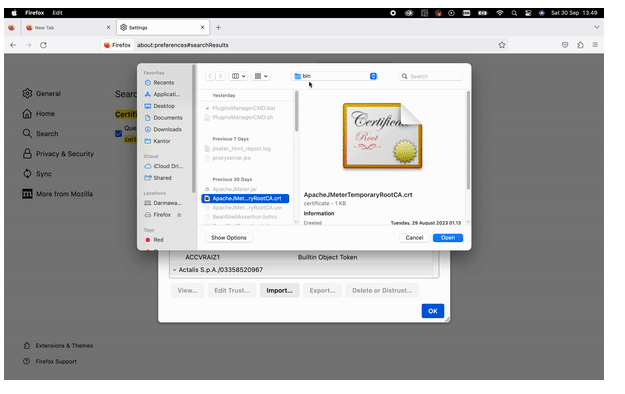
Description automatically generated

**2. Look for “Proxy settings” and select “Manual proxy configuration.”** Set the HTTP proxy to “localhost” and the port to “8888” (aligning with the JMeter HTTP(S) Test Script Recorder port settings). Ensure to check “Also use this proxy for HTTPS” before clicking ‘OK’.

A screenshot of a web browser

Description automatically generated

**3. Setting up SSL Certificate:**



* Navigate back to Firefox settings and search for “certificate.” Click on “View Certificates” and import the JMeter certificate found in “yourjmeterdirectorypath/bin/ApacheJMeterTemporaryRootCA.crt”.

4. **With the certificate imported**, your Firefox browser is now ready to record network interactions within your web application.

5. Return to JMeter, navigate to the HTTP(S) Test Script Recorder, and click on the ‘Start’ button. This activates the JMeter proxy to record all network traffic from your web application, crucial for subsequent API performance testing.

A screenshot of a computer

Description automatically generated

6. Open Firefox again and proceed to interact with your web application as you expect a user to. For instance, use a dummy login web application available at **https://practicetestautomation.com/practice-test-login/**.

A screenshot of a computer

Description automatically generated

7. Once the necessary interactions are recorded, stop the recorder within JMeter by using the Transaction Control.

A screenshot of a computer

Description automatically generated

8. All HTTP(S) requests made in your web application are now recorded and available under your ‘Login Test Recording Controller’ in JMeter. Feel free to remove unnecessary requests and rename them to enhance clarity in your Listener.

A screenshot of a computer program

Description automatically generated

9. Optionally, add assertions to each request, such as JSON assertion, HTML assertion, or Duration assertion, based on your testing requirements.

A screenshot of a computer

Description automatically generated

10. Adjust the ‘Number of Threads (users),’ ‘Ramp-up period,’ ‘Loop count,’ and other settings within your Test Group to align with your performance testing objectives.



11. With all configurations in place, run your test and observe the results in the Result Tree Listener under your test plan.

Below are the Api’s run mentioned on the LHS and summary report and their parameters measured.

A screenshot of a computer

Description automatically generated

Aggregate report

A screenshot of a computer

Description automatically generated

**Below are View results in tree structure**

A screenshot of a computer

Description automatically generated

**Below is the Results in Table format.**

A screenshot of a computer

Description automatically generated

By following these steps, you’ve effectively utilized Firefox in conjunction with JMeter to record HTTP(S) requests, paving the way for comprehensive API performance testing.

**Pro-Tips for Optimizing JMeter Performance Testing**

In the world of performance testing, every efficiency gained translates to a more accurate and insightful evaluation of your application’s capabilities. Here are some pro-tips to enhance your performance testing experience with JMeter:

**Utilize Terminal for Enhanced Efficiency:** When executing performance tests, opt for running JMeter from the terminal instead of using the GUI. This approach maximizes efficiency and optimizes resource usage, making it the preferred method for serious performance testing.

**Disable Listeners for Efficient Execution:** When running JMeter tests from the terminal, ensure that any unnecessary Listeners are disabled. Listeners can consume significant resources and impact the accuracy of performance results. By disabling them, you allocate resources more efficiently for the actual testing processes.

**Leverage Multiple Recording Controllers:** To enhance the versatility and thoroughness of your test scenarios, feel free to add multiple Recording Controllers within your Thread Group. This allows you to record various user interactions efficiently.

By implementing these pro-tips, you’ll maximize the potential of JMeter for performance testing, ensuring accurate results and efficient resource usage.

**Best Practices for Efficient Think Time Implementation**

**Realistic Think Time:** Aim for think time values that closely match the actual behavior of users within the application.

**Vary Think Time:** Introduce variability in think time by using random timers. This mimics the diverse behavior of users and enhances the accuracy of load simulation.

**Constant Monitoring and Adjustment**: Continuously monitor and adjust think time values based on application changes, user feedback, or performance tuning efforts.

**Conclusion: Incorporating think time during recording in JMeter is essential for accurate performance testing that mirrors real-world user behavior. By understanding the significance of think time and following best practices, you can enhance the effectiveness of your performance testing efforts, identify potential performance bottlenecks, and ensure a seamless user experience for your application. Remember, precise think time integration is the key to unlocking the full potential of JMeter in your performance testing endeavors. Happy testing!**

For more details copy/click paste the link [Apache JMeter - User's Manual: Getting Started](https://jmeter.apache.org/usermanual/get-started.html)