**Test Plan** window in JMeter is like a container which will contain all the elements required to perform a test which may be some listeners, graphs etc.

**Workbench** will keep elements on a temporary basis.

**Thread-Group** is the users that will be used to create or run the test.

**Latency** is known as the time of 1st byte received.

**Ramp-Up** period is known as after few seconds of interval a user will hitting the application.

**How to create test:** -

1. Start JMeter.
2. Create a test plan.
3. Create a thread group.
4. Add a Sampler (HTTP).
5. Add Listeners.
6. Run the test.

**Assertions** are the checks on the response received.

1. Response Assertion 2. Duration Assertion 3. Size Assertion 4. HTML Assertion 5. XPATH Assertion (mainly used in API Testing) 6. XML Assertion

**ERROR Threshold** means received HTML can handle error up to 38.

**Listeners** are the elements that gather information about the performance of test. It is used to view results of test.

1. View Results in Table 2. View Result Tree (Don’t use if load is heavy on application)
2. Aggregate Report
3. Graph Result (Don’t use if load is heavy on application)
4. Summary Report
5. Simple Data Writer (Use it because it dosen’t take more memory than UI reports)

**Tools available for recording JMeter UI Test:** - **BADBOY Software (where you can record UI elements and export it to JMeter) or Blazemeter (Chrome plugin)**

**Add Database in JMeter:** -

1. Add thread group in test plan
2. Add **configuration** parameter as **JDBC Database Connection**
3. Add jars in JMeter if necessary. (JMeter -> lib folder) and restart JMeter
4. Add JDBC request in thread group
5. Add Listeners in thread group
6. Run and validate the result
7. **Note: -** Variable Name must be present in JDBC Request and JDBC Connection Configuration and both name must be same.

**Create Assertions for Database:** -

1. Add **Response Assertion** in thread group
2. Add **Variables Names (Total columns in your sql result) of** JDBC Request (Created above)
3. Select **JMeter variable** in Response assertion (For Ex: - **col4\_2** i.e. it will correspond to column\_4 and row 2)
4. Add Listener **Assertion Result**  to check result of assertions
5. Run and Validate

**How to run JMeter from Command Line:** -

1. Go to JMeter -> bin folder
2. Command to execute test plan on cmd is **jmeter –n –t [location of jmeter test script] –l [location of the result file] ‘-n’ stand for non gui mode, ‘-t’ stands for location of jmeter file, ‘-l’ stands for location of result file**
3. Use **jmeter –h** on cmd for more options **or** use **jmeter -?** For more options

**Web Services and API in JMeter: -**

1. **REST API’s**: -
2. Add **HTTP Request Sampler**
3. Add REST API details in sampler
4. Run and Validate
5. Add Assertions if necessary to verify the result
6. **SOAP API’s:** -
7. Add **SOAP/XML – RPC Request**
8. Add entire **URL** in **URL** path of the request
9. In **Send SOAP Action** enter SOAPAction of that URL
10. Copy and Paste the XML Body of an API
11. Run and Validate
12. Add Assertions if necessary to verify the result

**Create HTML report using JMeter:** -

1. Create a test plan and save it
2. Go to **cmd** and change directory to JMeter -> bin
3. Command to execute test plan on cmd is **jmeter –n –t [location of jmeter test script] –l [location of the result file for ex: - csv] -e –o [location of output folder]**

**‘-n’ stand for non gui mode, ‘-t’ stands for location of jmeter file, ‘-l’ stands for location of result file**

1. Analyze HTML (Dashboard Reports)
2. You can create HTML report directly through csv file as below

**Jmeter –g [location of csv file] –o[location of output folder]**

How to use Plugin Manager: -

1. Download any plugin of JMeter
2. Put the .jar file in JMeter -> lib -> ext folder
3. Restart JMeter

**Functions in JMeter:** -

${\_\_functionName} is syntax of function in JMeter

${\_\_functionName (arguments to function)}

**functionNames** and **variableName** are case sensitive and naming must follow camelCasing

1. Log function: - ${\_\_log(“message”)} :- To print anything
2. Time function: - ${\_\_time(dd MM YYYY HH mm ss)}:- To display time
3. ThreadNum function: - ${\_\_threadNum}: - To display thread number
4. intSum: - ${\_\_intSum(2,3)}:- It will add passed numbers
5. intSum: - ${\_\_intSum(2,3),result}:- It will add passed numbers and stored the result in variable **result.** We can use this variables as **${result}**

Like these we have so many functions to be used in JMeter. For more options go to **Tools -> Function Helper Dialog** or we can get it from google also.

**Realistic Performance Test – PACING: -**

**Think Time: -** Simulate actual user action with timing/delays

**Pacing:** - To controlled ramp-up and down of virtual users, control timing between iterations and achieve x interactions in y min/sec

1. Add plugin **Stepping Thread**
2. Set all the details appropriately

**Timers** helps to achieve realistic performance. To pause thread for some time, to add delay between threads, to avoid over flooding the server and achieve real time behavior by pacing the load.

99% of request can be controlled by **Constant Timer and Uniform Random Timer**

**How to use Regular Expression Extractor for Correlation:** -

1. Create a test plan
2. Add **Regular Expression Extractor** from Post Processor
3. Add the details
4. Run and validate

**JMeter Templates:** -

Reusable project scripts known as templates.

1. Save your test plan as .jmx file
2. Place the .jmx inside JMeter ->bin->templates folder
3. Edit templates.xml file as below in <templates> tag

<template isTestPlan=”true”>

<name>Recording</name>

<filename>/bin/templates/recording.jmx</filename>

<description>Some Description Here</description>

</template>

1. Restart JMeter

**How to record login test in JMeter: -**

1. Add **Blazemeter** plugin to chorme
2. Start blazemeter and login to it
3. Start and Stop recording and export it to .jmx file
4. Import jmx file in JMeter
5. Add Listeners to view the report

**How to Monitor Server Health: - (By using PERFMON)**

1. Add **PERFMON** plugin in JMeter either from Google or from Plugin Manager in JMeter
2. Download **ServerAgent zip** from google and start Perfmon agent (click on bat file)
3. Create JMeter test to monitor server health
4. Add **Perform Metric Collector** Listener in thread group. Add Servers/Monitor Details
5. Add **HTTP Request** for performing actions on server and **View Result Tree** listener for result.

**JMeter Remote Testing (Distributed Testing):** -

1. SetUp Master
2. Create slaves (goto JMeter -> bin -> jmeter.properties -> add remote\_hosts=<ip\_address of machine>)
3. Create a keystore file (goto JMeter -> bin -> createkeystore.bat file)
4. While creating keystore file please note following points: -

First and last name should be **rmi**

Password should be **changeit**

1. After this a new file **rmRui\_keystore.jks** will be created in bin folder
2. Restart JMeter
3. Run **jmeter-server** file on all slave(remote) machines
4. Go to Run->Remote Start ->Select the proper ip (previously added in properties file)
5. From **CMD** run as **jmeter –n –t [location of jmx file] –o [location of result file] –R [ip of slave machine]**

**JMeter – Selenium Integration:** -

1. Add **Selenium support plugin and restart jmeter**
2. Add **Chromedriver config and Webdriver Sampler to thread group**
3. Give chromedriver.exe path in chromedriver config
4. Add scripts in webdriver sampler
5. Run and Validate

**JMX Check Tool:** -

1. Download and install jmx checker tool (Test Plan Checker )in Jmeter
2. Goto **cmd** and cd till bin folder and enter command as

**TestPlanCheck.bat –jmx [path of jmx file]**

1. For more option we can try: -

**TestPlanCheck.bat –jmx [path of jmx file] –stats / TestPlanCheck.bat –jmx [path of jmx file]**

**--tree-dump**