



**Fresher Academy**



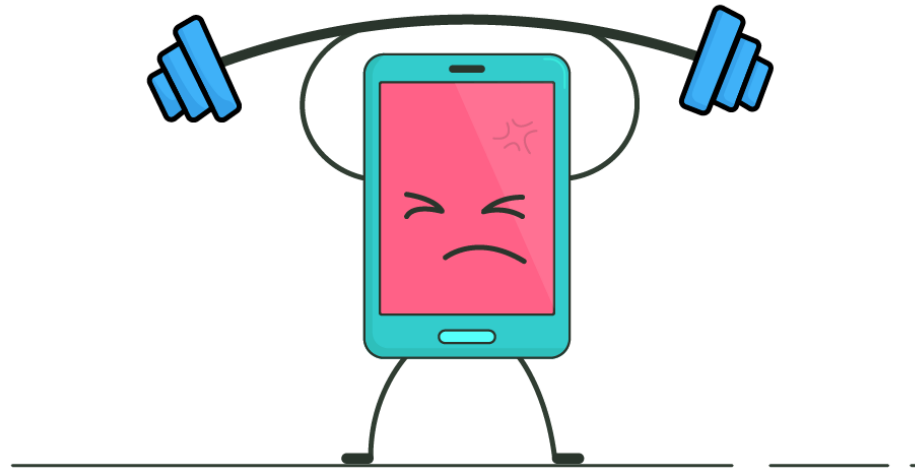
# Jmeter Basics & Setup and Understand JMeter Environment

# Agenda

- Performance Testing Introduction
- Types of Performance Testing
- What is Load Testing?
- When should you use Load Testing?
- What is Stress Testing?
- When Should You Use Stress Testing?
- Jmeter Introduction
- Test Plan, Thread Group
- Different type of steps in Jmeter
- Execution order of Elements
- Simple Controller
- Random Order Controller
- Interleave Controller
- Loop Controller
- If Controller
- Only Once Controller
- Runtime Controller
- Module Controller

# Performance Testing Introduction

**Performance testing** is a type of testing intended to determine the responsiveness, reliability, throughput, interoperability, and scalability of a system and/or application under a given workload.



# Performance Testing Introduction

The focus of Performance Testing is checking a software program's

**Speed** - Determines whether the application responds quickly

**Scalability** - Determines maximum user load the software application can handle.

**Stability** - Determines if the application is stable under varying loads

# Types of Performance Testing

**Load testing** - checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.

Test số lượng user trong 1 khoảng tg nhất định (tăng dần số lượng user)

**Stress testing** - involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.

Test để tìm ngưỡng của hệ thống (test với số lượng user lớn ngay từ khi bắt đầu)

**Endurance testing** - is done to make sure the software can handle the expected load over a long period of time.

**Spike testing** - tests the software's reaction to sudden large spikes in the load generated by users.

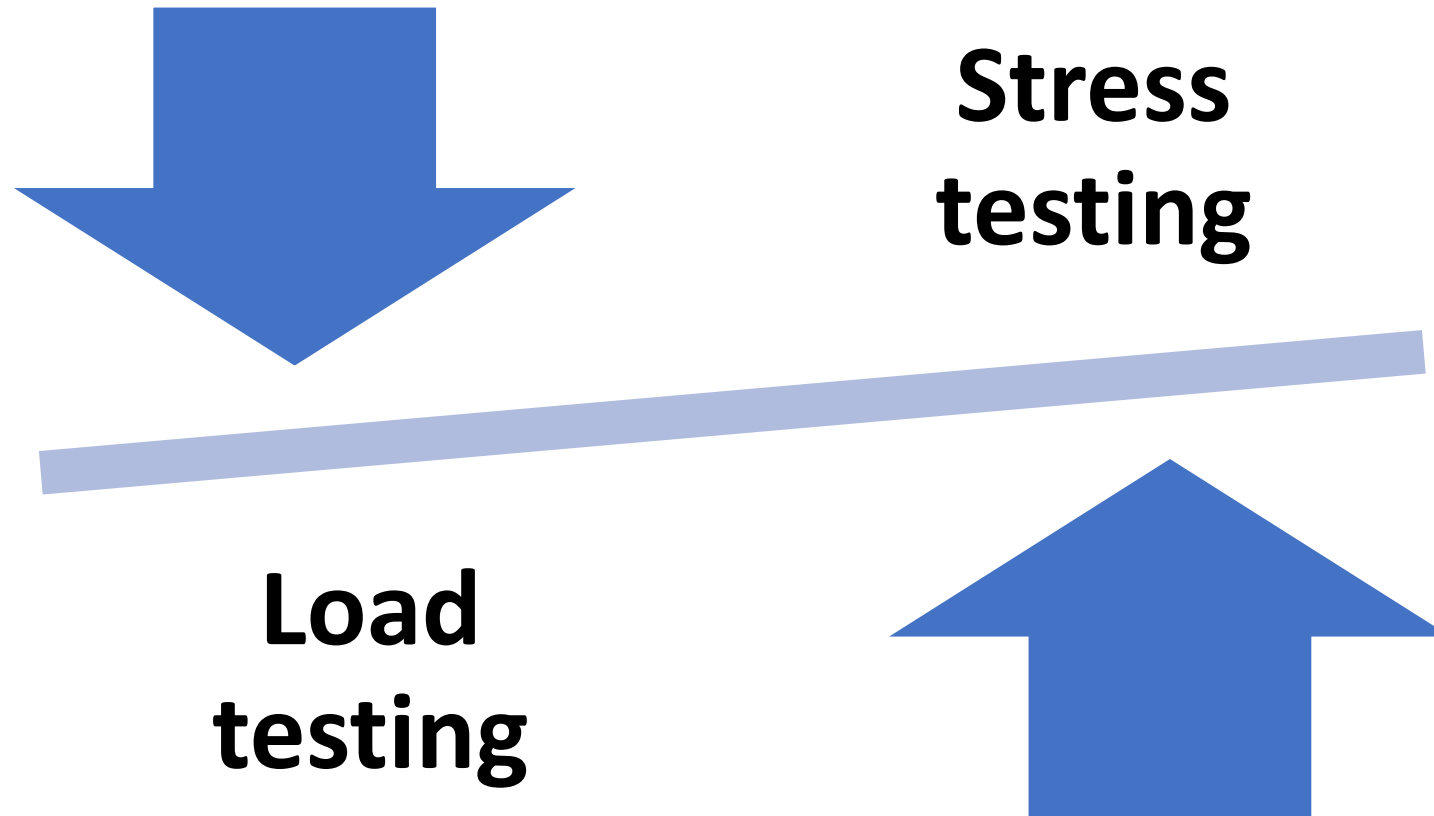
Test với điều kiện đột biến: vd từ 10 user → 1000 user

**Volume testing** - Under Volume Testing large no. of. Data is populated in a database and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.

**Scalability testing** - The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.

Dự đoán về sự gia tăng số lượng user để tính toán sự thêm vào của phần cứng

# Types of Performance Testing



# What is Load Testing?

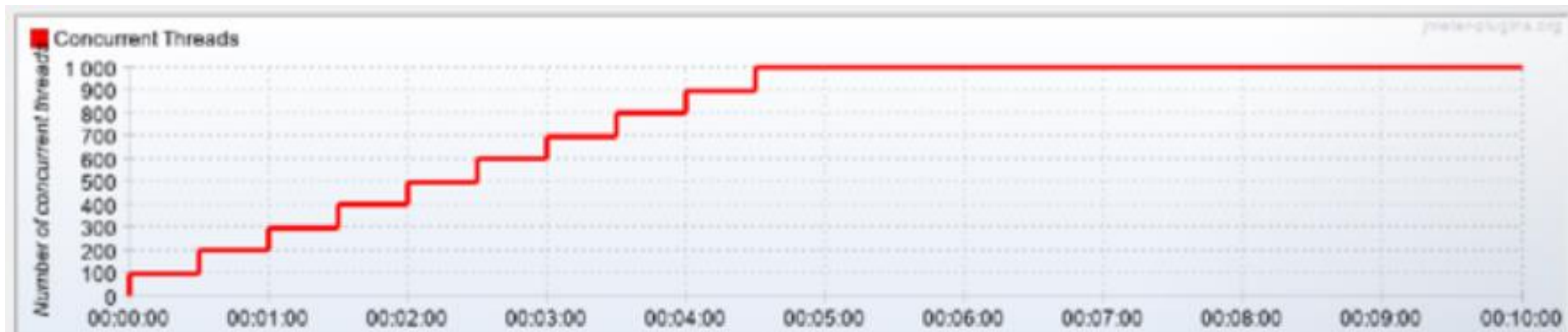
**Load testing** is testing that checks how systems function under a heavy number of concurrent virtual users performing transactions over a certain period of time. Or in other words, how systems handle heavy load volumes.



# When should you use Load Testing?

When you want to determine how many users your system can handle. You can determine different user scenarios that let you focus on different parts of your system, like the checkout webpage on your website or app for web load testing.

This test analyzes adding **100** users every **30** seconds until reaching **1,000** users. The entire stepping process takes **300** seconds. After reaching **1,000** threads all of them will continue running and hitting the server together for **5** minutes.





# What is Stress Testing?

**Stress testing** is testing that checks the upper limits of your system by testing it under extreme loads.



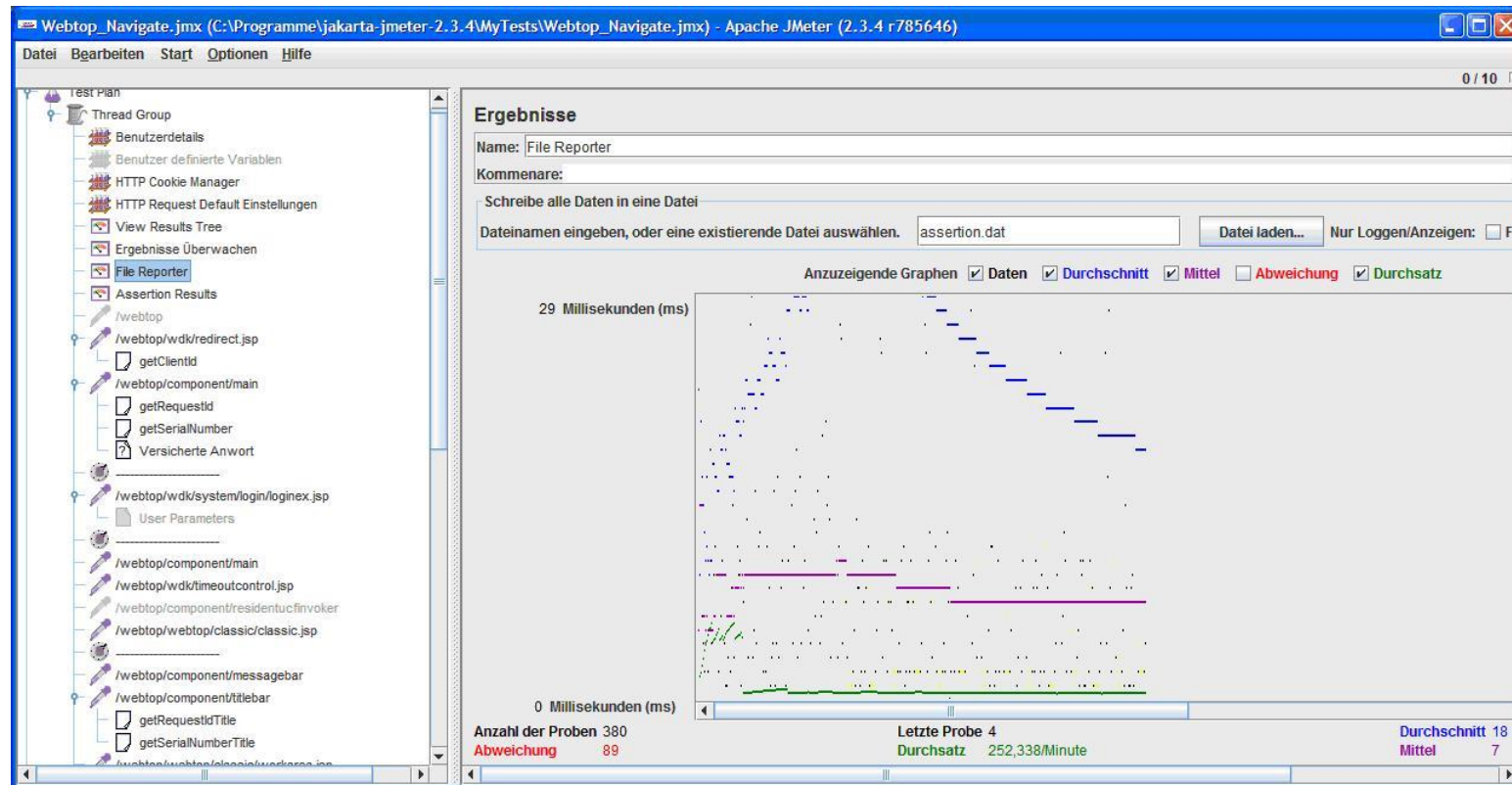
# When Should You Use Stress Testing?

Website stress tests and app stress tests are important before major events



# Jmeter Introduction

**JMeter** is a desktop application, designed to test and measure the performance and functional behavior of client/server applications, such as web applications or FTP applications.



# Jmeter Introduction - Advantages

Open source license

Friendly GUI

Platform independent

Full multi-threading  
framework

Visualize Test Result

Easy installation

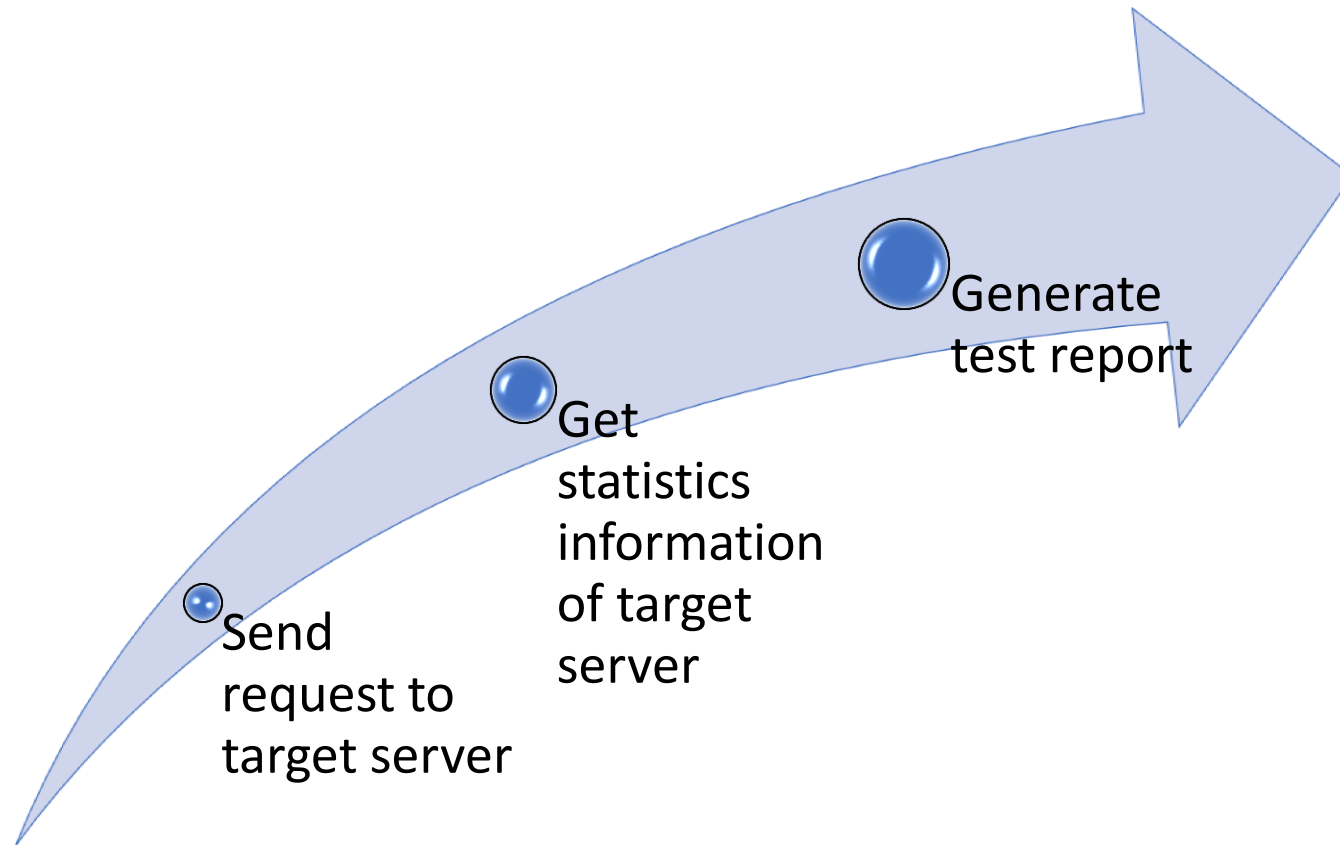
Highly extensible

Unlimited testing  
capabilities

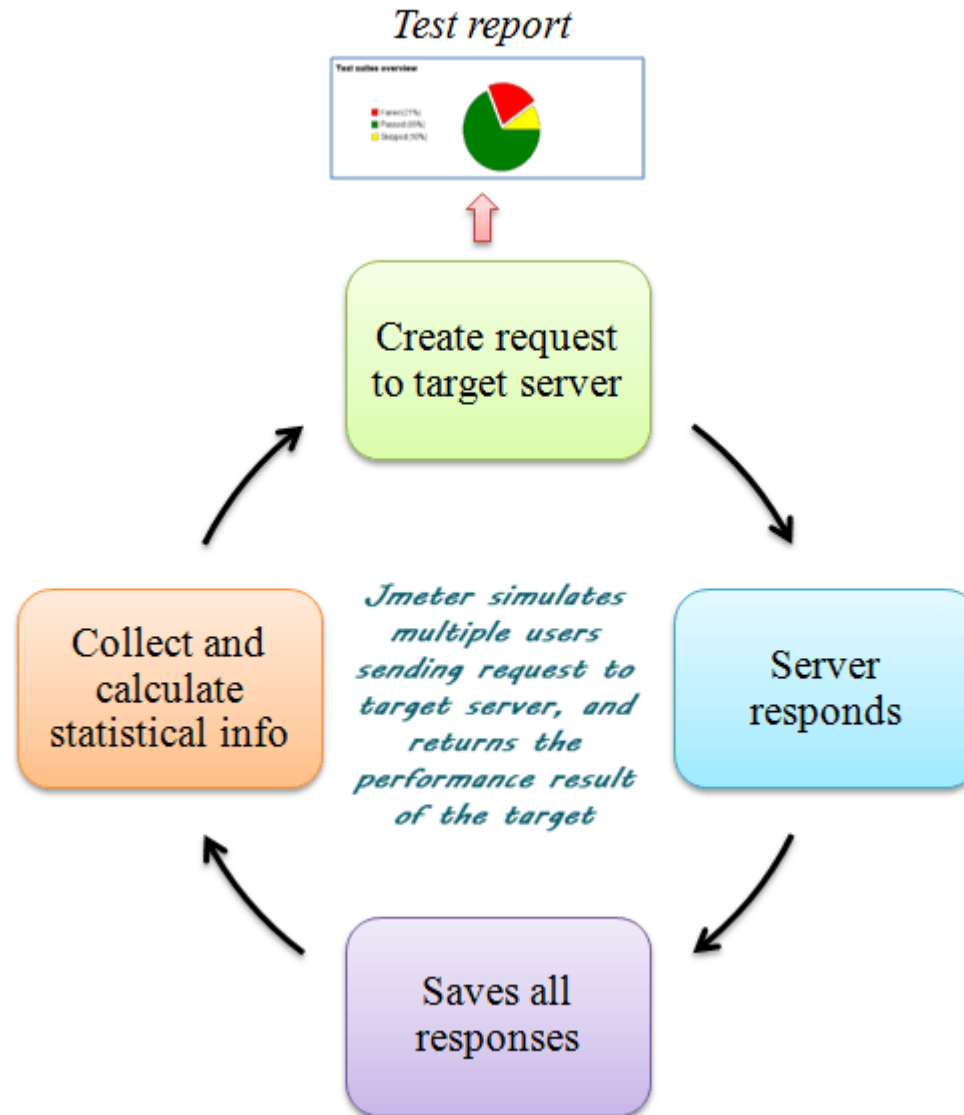
Support multi protocol

# Jmeter Introduction - JMeter work

**JMeter** simulates a group of users sending requests to a target server, and return statistics information of target server through graphical diagrams

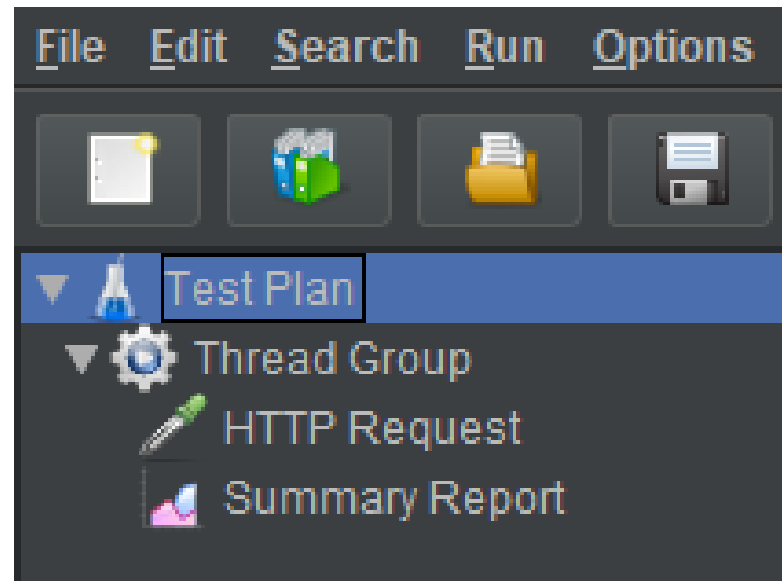
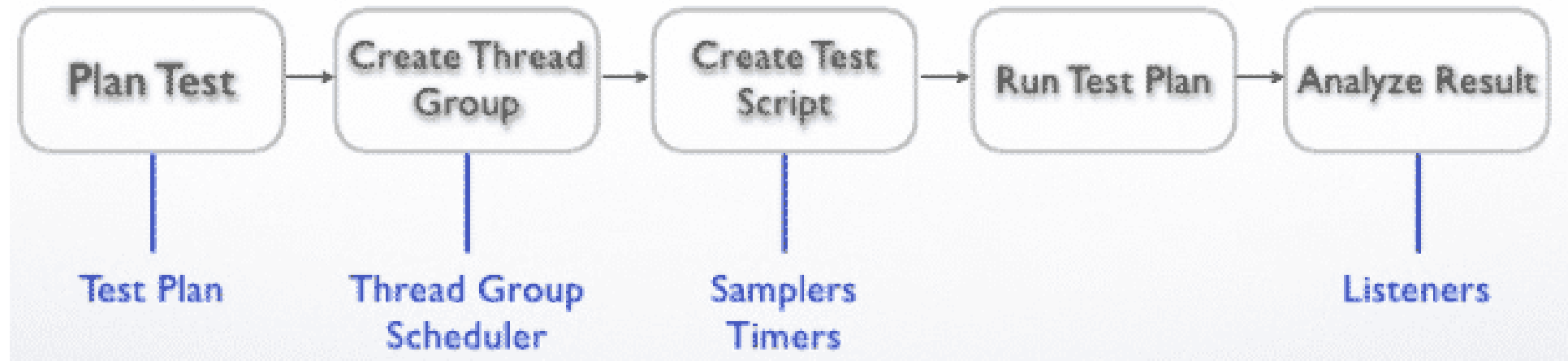


# Jmeter Introduction - JMeter work



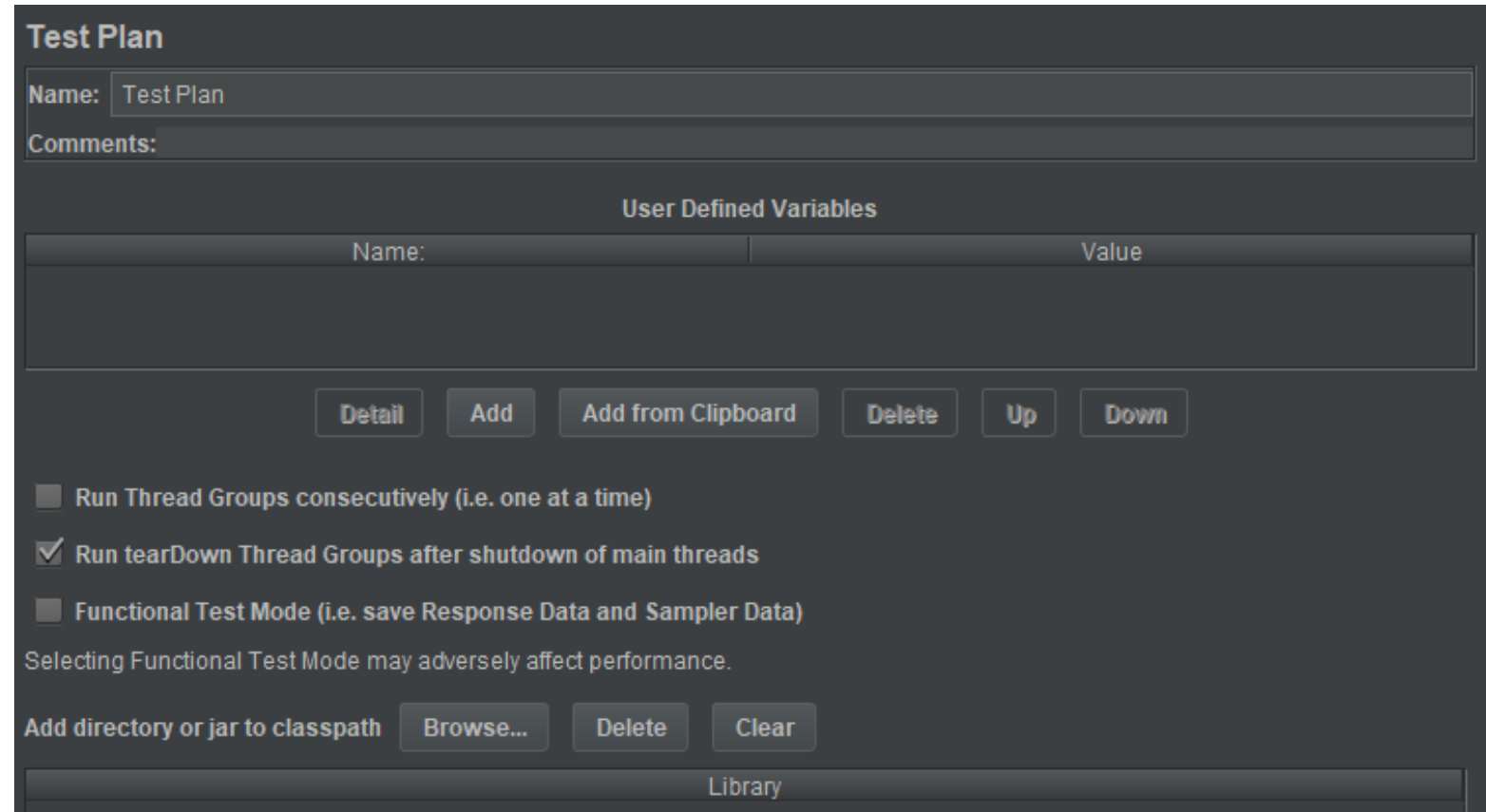
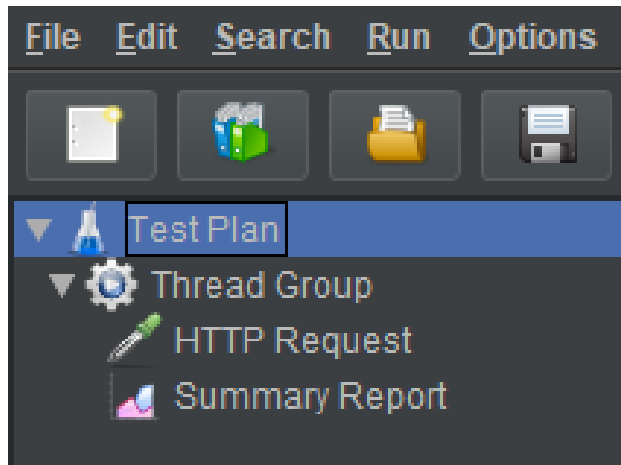


# Test Plan, Thread Group



# Test Plan, Thread Group

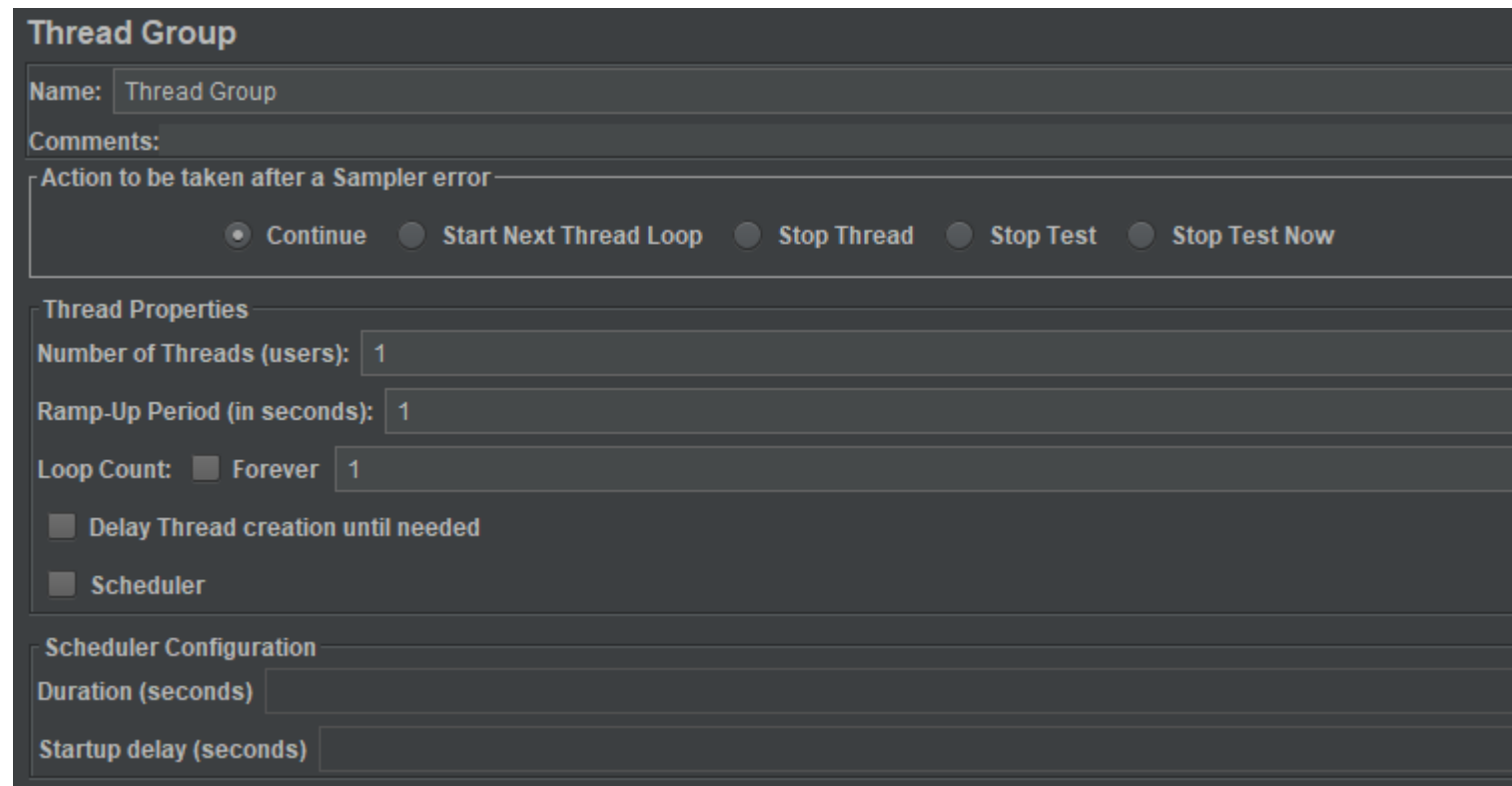
**Test Plan** is the root element of the JMeter scripts and houses the other components, such as Threads, Config Elements, Timers, Pre-Processors, Post-Processors, Assertions, and Listeners.





# Test Plan, Thread Group

**Thread group**—These elements are used to specify number of running threads, a ramp-up period, and loop-count (no. of times to execute the test). Each thread simulates a user and the ramp-up period specifies the time to create all the threads.



The screenshot shows the 'Thread Group' configuration window in JMeter. It includes fields for Name, Comments, and Action to be taken after a Sampler error. The Thread Properties section contains fields for Number of Threads (users), Ramp-Up Period (in seconds), Loop Count, and checkboxes for Delay Thread creation until needed and Scheduler. The Scheduler Configuration section includes fields for Duration (seconds) and Startup delay (seconds).

**Thread Group**

Name: Thread Group

Comments:

Action to be taken after a Sampler error

☒ Continue ☐ Start Next Thread Loop ☐ Stop Thread ☐ Stop Test ☐ Stop Test Now

**Thread Properties**

Number of Threads (users): 1

Ramp-Up Period (in seconds): 1

Loop Count: ☐ Forever 1

☐ Delay Thread creation until needed

☐ Scheduler

**Scheduler Configuration**

Duration (seconds)

Startup delay (seconds)

# Different type of steps in Jmeter

Controllers

Listeners

Timers

Assertions

Configuration  
Elements

Pre-Processor  
Elements

Post-  
Processor  
Elements

# Different type of steps in Jmeter - Controllers

## Samplers

- These allow JMeter to send specific types of requests to a server. In our sample tests later, we will be sending HTTP Requests the most, so we will use HTTP Request Sampler to let JMeter send those requests. You may add Configuration Elements to these Samplers to customize your server requests.

## Logic Controllers

- These allow you to customize the logic that JMeter uses to decide when to send requests. For example, you can use Random Controllers to send HTTP requests to the server randomly.

# Different type of steps in Jmeter - Controllers

## Samplers

- HTTP Request
- FTP Request
- JDBC Request
- Java Request
- SOAP/XML-RPC Request
- WebService (SOAP) Request
- LDAP Request
- LDAP Extended Request
- Access Log Sampler
- BeanShell Sampler
- BSF Sampler
- TCP Sampler
- JMS Publisher
- JMS Subscriber
- JMS Point-to-Point
- JUnit Request
- Mail Reader Sampler
- Test Action

## Logic Controllers

- Simple Controller
- Loop Controller
- Once Only Controller
- Interleave Controller
- Random Controller
- Random Order Controller
- Throughput Controller
- Runtime Controller
- If Controller
- While Controller
- Switch Controller
- ForEach Controller
- Module Controller
- Include Controller
- Transaction Controller
- Recording Controller

# Different type of steps in Jmeter - Listeners

**Listeners** let you view the results of the Samplers in the form of tables, graphs, trees or simple text in some log files. They provide visual access to the data gathered by JMeter about the test cases as a Sampler component of JMeter is executed.

The screenshot shows the JMeter 'Aggregate Graph' listener configuration window. At the top, the title is 'Aggregate Graph'. Below it, there are fields for 'Name' (set to 'Aggregate Graph') and 'Comments'. A section titled 'Write results to file / Read from file' contains a 'Filename' input field, a 'Browse...' button, and a 'Log/Display On' button. Below this is a table showing test results. The table has columns: Label, # Samples, Average, Median, 90% Line, 95% Line, 99% Line, Min, Maximum, and Error. The first row is labeled 'TOTAL' and shows values: 0, 0, 0, 0, 0, 0, 0, 922337203..., -922337203..., and an empty error cell. Below the table, there are tabs for 'Settings' and 'Graph'. The 'Settings' tab is active, showing a 'Display Graph' button and a 'Save Graph' button. At the bottom, there is a 'Column settings' section with 'Columns to display' and 'Value font' options. The 'Columns to display' section has checkboxes and colored boxes for 'Average' (red), 'Median' (blue), '90% Line' (green), '95% Line' (yellow), '99% Line' (purple), and 'Min'. The 'Value font' section has dropdowns for 'Sans Serif', 'Size: 10', 'Style: Normal', and checkboxes for 'Draw outlines bar?', 'Show number grouping?', and 'Value label'.

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error
TOTAL	0	0	0	0	0	0	922337203...	-922337203...	

# Different type of steps in Jmeter - Listeners

Sample Result Save Configuration

Graph Full Results

Graph Results

Spline Visualizer

Assertion Results

View Results Tree

Aggregate Report

View Results in Table

Simple Data Writer

Monitor Results

Distribution Graph (alpha)

Aggregate Graph

Mailer Visualizer

BeanShell Listener

Summary Report

# Different type of steps in Jmeter - Timers

**A Timer Component** is an option in building a Test Plan. It causes JMeter to pause for a certain amount of time between two successive requests that a Thread Group makes.

Constant Timer

Gaussian Random Timer

Uniform Random Timer

Constant Throughput Timer

Synchronizing Timer

BeanShell Time

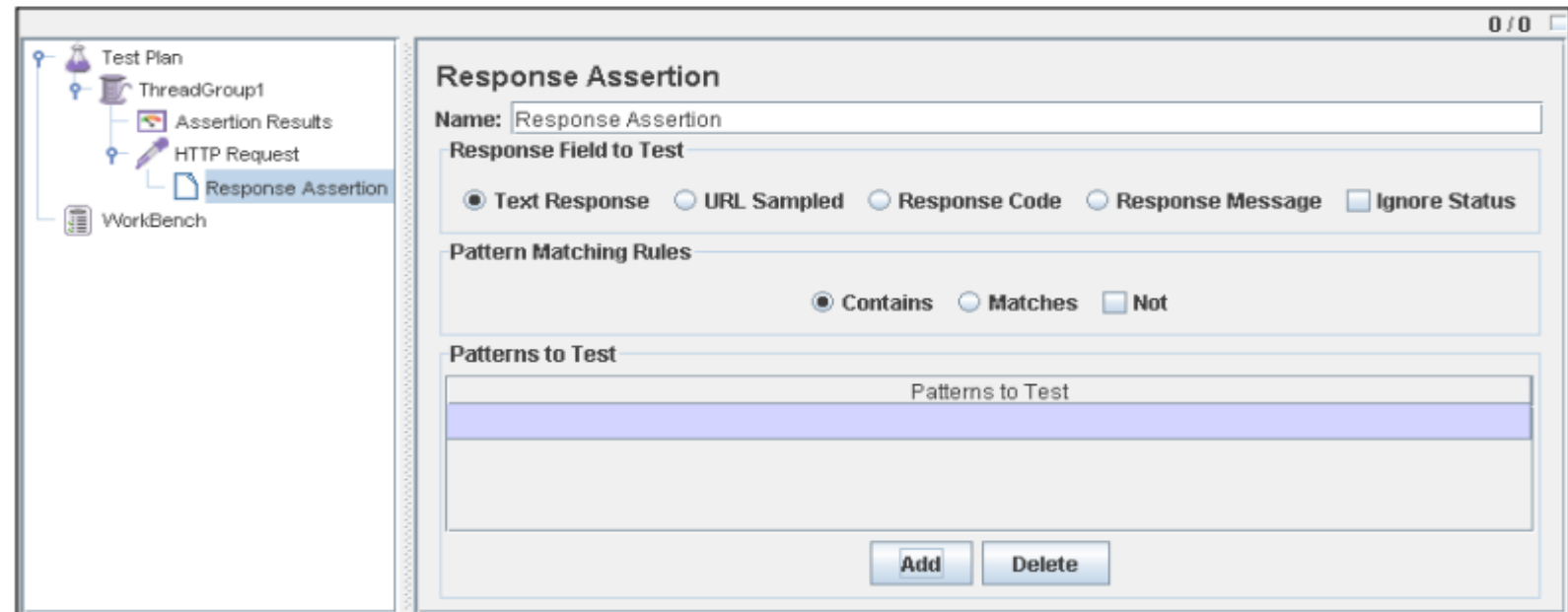


# Different type of steps in Jmeter - Assertions

**Assertions** allow you to include some validation test on the response of your request made using a Sampler. They are inserted as a child component of a Sampler.

With Assertion, you can assert whether the application is returning the expected result or not. JMeter allows you to specify your assertions using Perl-style regular expressions.

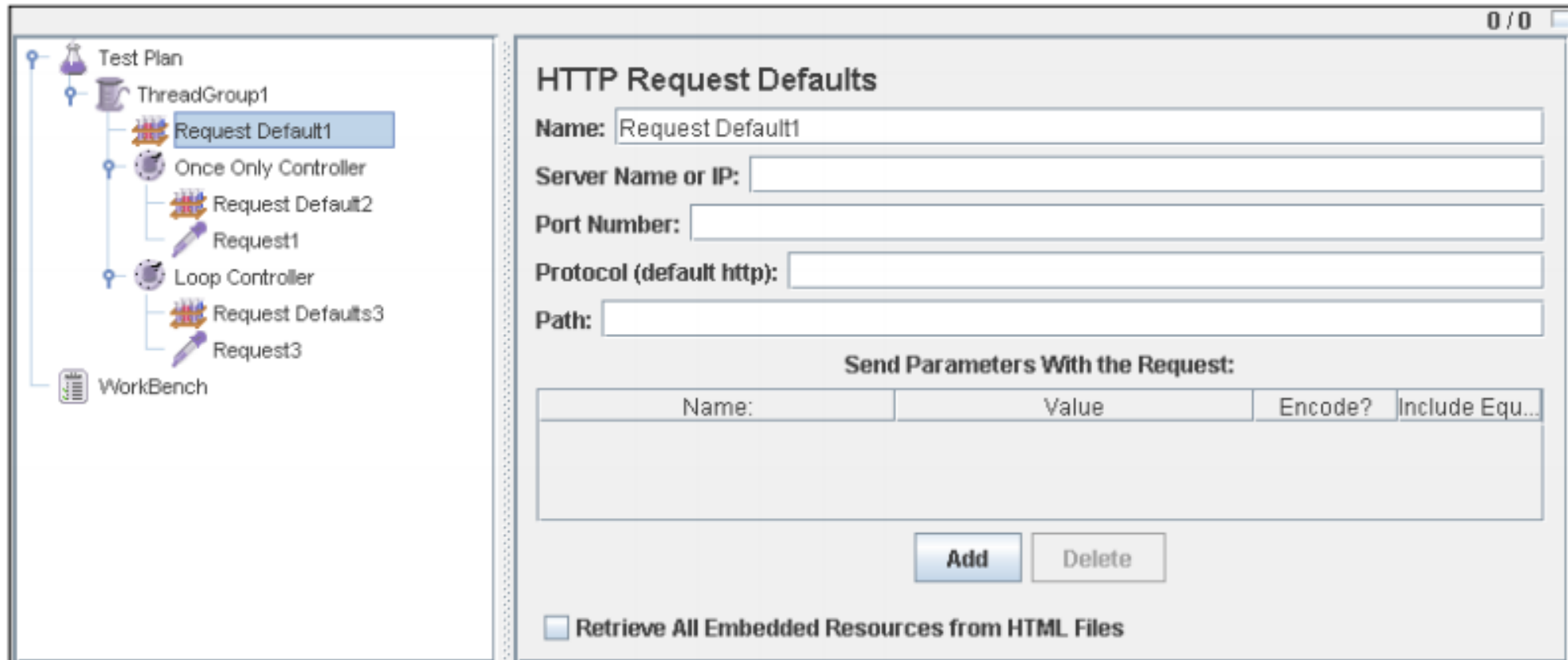
- Response Assertion
- Duration Assertion
- Size Assertion
- XML Assertion
- BeanShell Assertion
- MD5Hex Assertion
- HTML Assertion
- XPath Assertion
- XML Schema Assertion





# Different type of steps in Jmeter - Configuration Elements

**Configuration Elements** allow you to create defaults and variables to be used by Samplers. They are used to add or modify requests made by Samplers.



# Different type of steps in Jmeter - Configuration Elements

- CSV Data Set Config
- FTP Request Defaults
- HTTP Authorization Manager
- HTTP Cookie Manager
- HTTP Proxy Server
- HTTP Request Defaults
- HTTP Header Manager
- Java Request Defaults
- JDBC Connection Configuration
- Login Config Element
- LDAP Request Defaults
- LDAP Extended Request Defaults
- TCP Sampler Config
- User Defined Variables
- Simple Config Element

# Different type of steps in Jmeter - Pre-Processor Elements

**Pre-processors** allow you to modify the Samplers in their scope. They are often used to modify the settings of a Sample Request just before it runs, or to update variables that are not extracted from response text.

HTML Link Parser

HTTP URL Re-writing Modifier

HTML Parameter Mask

HTTP User Parameter Modifier

User Parameters

Counter

BeanShell PreProcessor

# Different type of steps in Jmeter - Post-Processor Elements

**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

Regular Expression Extractor

XPath Extractor

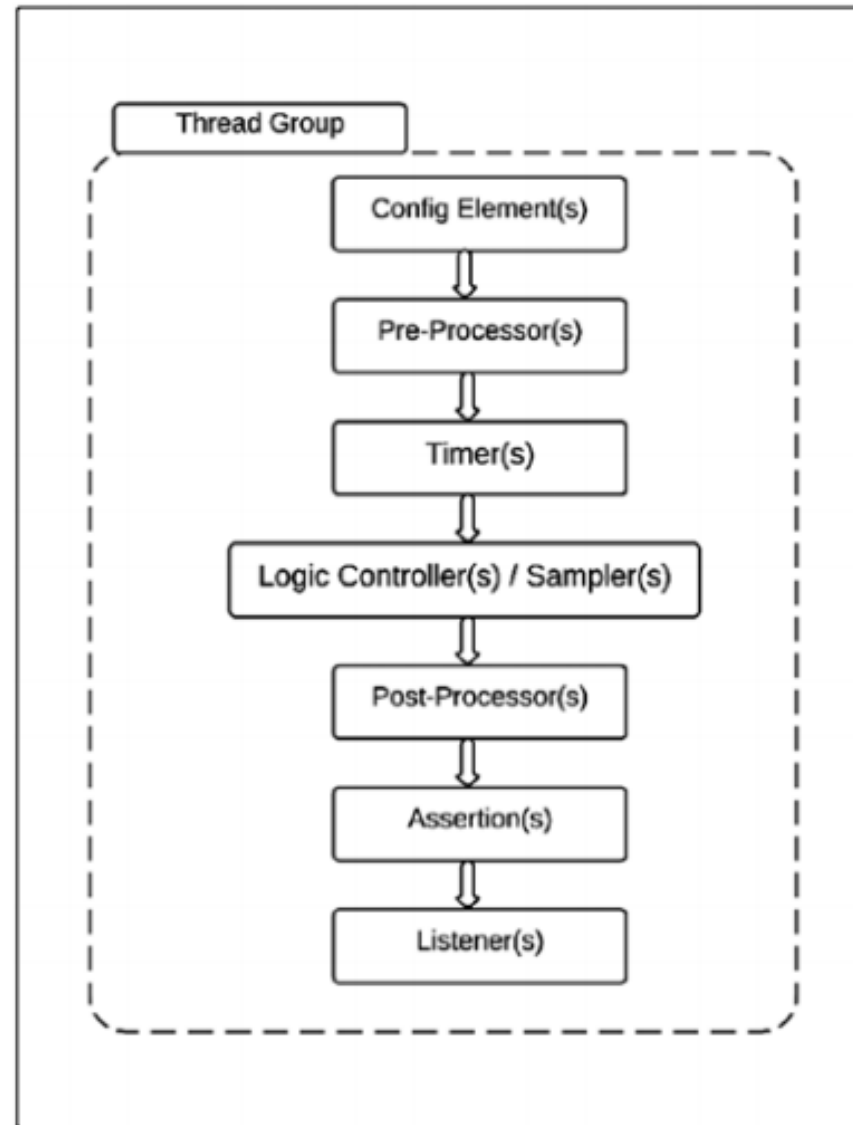
Result Status Action Handler

Save Responses to a file

Generate Summary Results

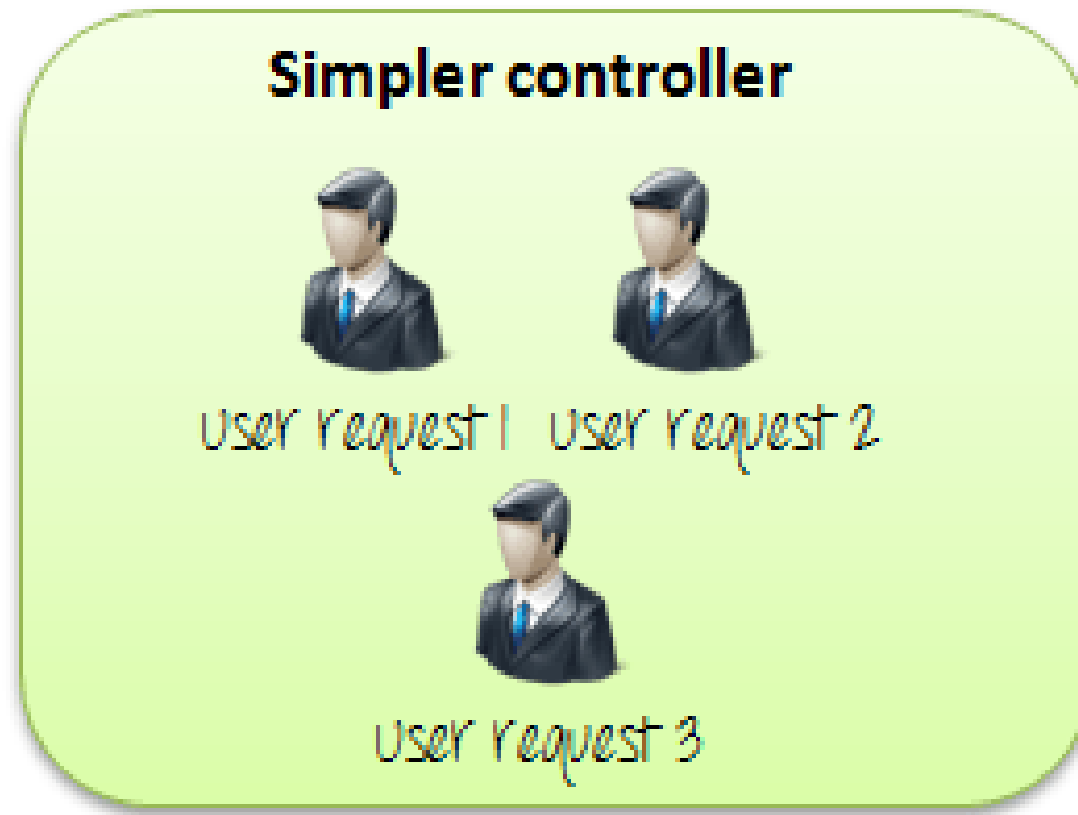
BeanShell PostProcessor

# Execution order of Elements



# Simple Controller

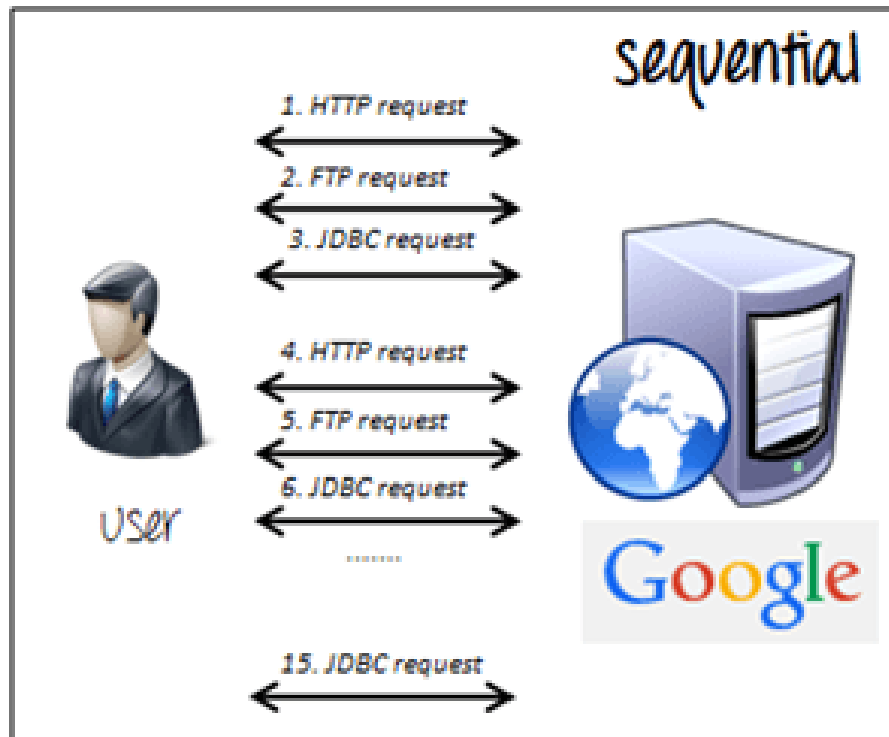
**Simple Controller** is just a container for user request.



# Random Order Controller

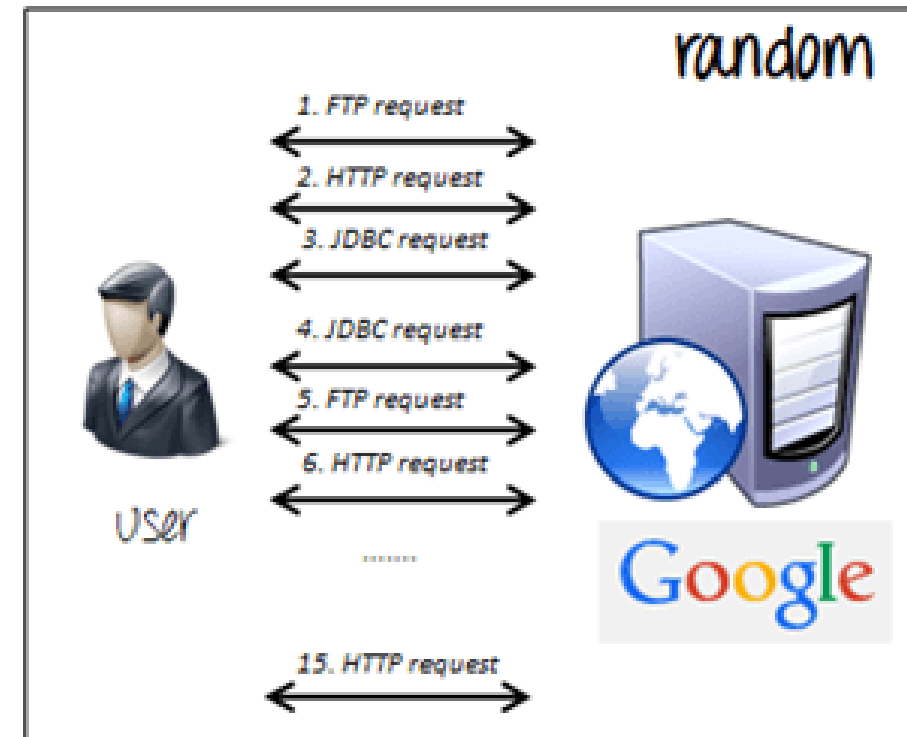
**Random Controller** makes all the user requests run in **the random** order in each loop period.

HTTP request  
FTP request  
JDBC request



All request send in sequential order

VS

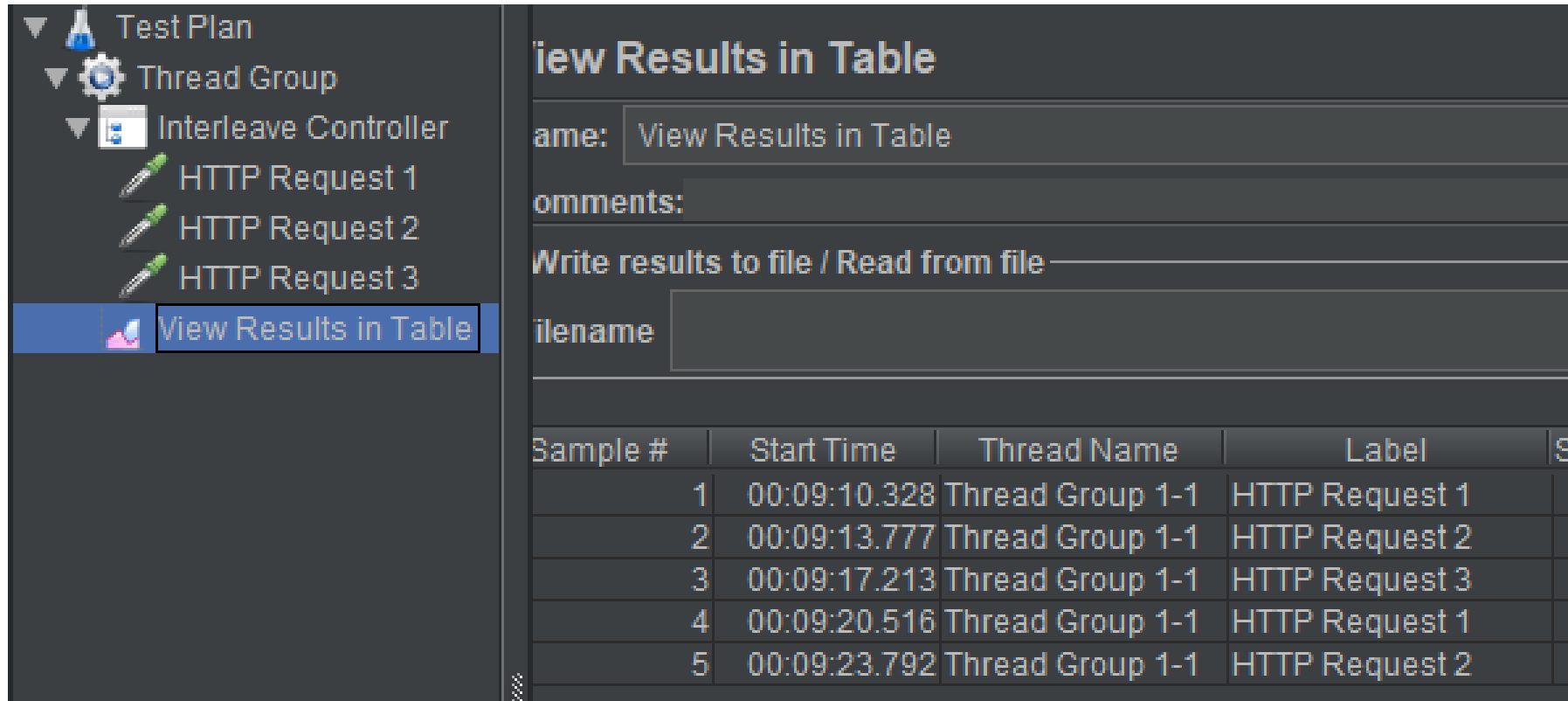


All request send in random order

# Interleave Controller

**Interleave Controller** will make one of samplers contained in it run in each loop of the thread and but the samplers in each loop will be in sequential order.

The thread-group has a loop-count of **5** and the number of threads as **1**



The screenshot displays the JMeter GUI. On the left, the Test Plan tree shows a Thread Group containing an Interleave Controller with three HTTP Request samplers (HTTP Request 1, HTTP Request 2, HTTP Request 3) and a View Results in Table sampler. The View Results in Table sampler is selected. On the right, the configuration for 'View Results in Table' is shown, including fields for Name, Comments, and a checkbox for 'Write results to file / Read from file'. Below these fields is a table displaying the results of the test run.

Sample #	Start Time	Thread Name	Label	S
1	00:09:10.328	Thread Group 1-1	HTTP Request 1	
2	00:09:13.777	Thread Group 1-1	HTTP Request 2	
3	00:09:17.213	Thread Group 1-1	HTTP Request 3	
4	00:09:20.516	Thread Group 1-1	HTTP Request 1	
5	00:09:23.792	Thread Group 1-1	HTTP Request 2	



# Loop Controller

**The Loop Controller** will make the contained samplers run as many times as the loop count or forever if the forever check-box is checked.

**Loop Controller**

**Name:** Loop Controller

**Comments:**

**Loop Count:** ☐ Forever

Start Time	Thread Name	Label	S
00:12:25.474	Thread Group 1-1	HTTP Request 1	
00:12:28.793	Thread Group 1-1	HTTP Request 2	
00:12:32.167	Thread Group 1-1	HTTP Request 3	
00:12:35.635	Thread Group 1-1	HTTP Request 1	
00:12:39.053	Thread Group 1-1	HTTP Request 2	
00:12:42.761	Thread Group 1-1	HTTP Request 3	

# If Controller

**If controller** in jmeter allows you to set condition to evaluate it and based on condition evaluation result it will decide to run or not to run if controller's child elements.

**If Controller**

Name: If Controller

Comments:

For performance it is advised to check "Interpret Condition as Variable Expression" and use `__jexl3` or `__groovy` evaluating to true or false or a variable that contains true  
`${JMeterThread.last_sample_ok}` can be used to test if last sampler was successful

1 `true`

Expression (must evaluate to true or false)

☒ Interpret Condition as Variable Expression? ☐ Evaluate for all children?

# If Controller



**Interpret Condition as Variable Expression?**



**Evaluate for all children?**

## Interpret Condition as Variable Expression

If this is selected, then the condition must be an expression that evaluates to "true" (case is ignored).

Example, `${FOUND}` or `${_jexl(${VAR} > 100)}`. Unlike the Javascript case, the condition is only checked to see if it matches "true" (case is ignored).

## Evaluate for All Children

Should the condition be evaluated for all children? If not checked, then the condition is only evaluated on entry.

# If Controller

## Simple Conditions

- `1==1`: always evaluates to true,
- `"${JMeterThread.last_sample_ok}"`: checks if the last request was successful,
- `"${my_variable}" == "1"`: checks if `my_variable` variable is equal to 1,
- `${__jexl3("${COUNT}" == "1", )}`: checks if `COUNT` variable is equal to 1 using Jexl3,
- `${__groovy("${COUNT}" == "1")}`: checks if `COUNT` variable is equal to 1 using Groovy.

# If Controller

## Comparison Operators

- `==`: Left value must be equal to right value,
- `!=`: Left value must be different than right value,
- `<=`: Left value must be inferior or equal to right value,
- `>=`: Left value must be superior or equal to right value,
- `<`: Left value must be inferior to right value,
- `>`: Left value must be superior to right value.

# Only Once Controller

**The Only Once Controller** will make its child samplers run only once per thread i.e. only in the first loop.

**Thread Properties**

Number of Threads (users): 1

Ramp-Up Period (in seconds): 1

Loop Count: ☐ Forever 2

☐ Delay Thread creation until needed

☐ Scheduler

Once Only Controller

- HTTP Request 1
- HTTP Request 2
- HTTP Request 3
- View Results in Table

from file

Thread Name	Label
Thread Group 1-1	HTTP Request 1
Thread Group 1-1	HTTP Request 2
Thread Group 1-1	HTTP Request 3

**Runtime Controller** controls the execution of its samplers/requests for the given time.

# Runtime Controller

Name: Runtime Controller

Comments:

Runtime (seconds) 5

Runtime Controller

HTTP Request 1

HTTP Request 2

HTTP Request 3

View Results in Table

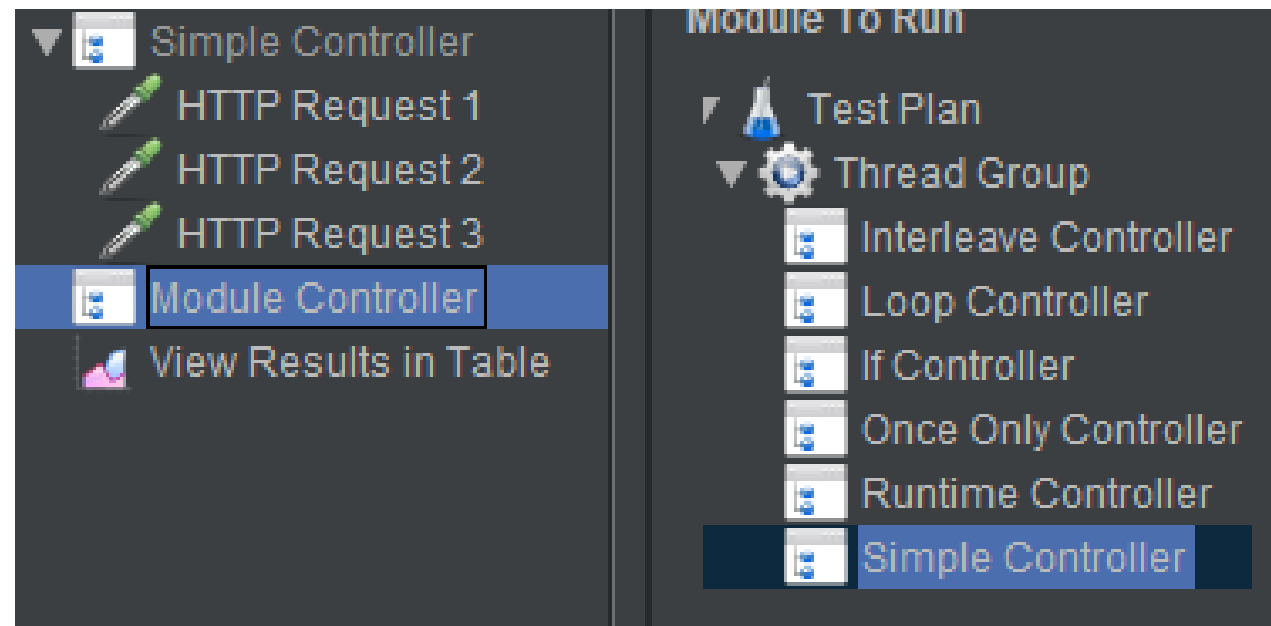
	Start Time	Thread Name	Label	State
1	00:27:35.459	Thread Group 1-1	HTTP Request 1	
2	00:27:39.090	Thread Group 1-1	HTTP Request 2	

# Module Controller

**Module controller** adds modularity to the JMeter Test Plan. Normally we construct test plan consists of small units of functionality like Login, Homepage, Logout.

Start Time	Thread Name	Label	Sample Time(...)
00:49:34.172	Thread Group 1-1	HTTP Request 1	3779
00:49:37.952	Thread Group 1-1	HTTP Request 2	3322
00:49:41.275	Thread Group 1-1	HTTP Request 3	3371

Chỉ muốn chạy 1 số Request nào đó thì ném vào Module controller thay vì phải disable những Request còn lại







# Fresher Academy



**Happy Coding!**