

# JMeter



Fernando Gómez, Javier Acosta, Matías García Marset, Carolina Wright

# JMeter - La herramienta

- Proyecto de Apache
- Open source, escrito en Java
- Es una herramienta de carga
- Inicialmente diseñada para pruebas de estrés
- Diseño y automatización de pruebas funcionales
- Se pueden testear los siguientes tipos de interfaces: HTTP, HTTPS, SOAP, FTP, LDAP, POP3, IMAP, SMTP, JMS, JDBC Y TPC
- Simula las funcionalidades de un navegador, o de cualquier cliente, siendo capaz de manipular los resultados y reutilizarlos
- Provee una interfaz para construir los tests

# JMeter - Test plan

El test plan es un script en formato XML (\*.jmx), generado por la interfaz de JMeter. Describe los pasos a ejecutar. Dentro de un test plan se puede configurar:

- Threads
- Config element
- Listener
- Timer
- Pre/Post processors
- Assertions
- Test fragment
- Non-test Elements

# Test plan - Threads

- En el test plan se especifica el número de threads
- Cada thread representa un usuario, que realiza las interacciones que se especifican en el test plan
- Los thread se pueden agrupar (*Thread group*). Todos los usuarios dentro de un mismo grupo realizan las mismas interacciones con la aplicación.
- Las interacciones de los usuarios con la aplicación se denominan *samplers*.
- Por cada ejecución se captura: timestamp, URL, resultado, tiempo de respuesta.
- Se permite agregar lógica entre cada sampler, como cuando ejecutarlo, hacer loops, etc con una herramienta llamada controller.

# Test plan - Threads

## 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

Tiempo en iniciarse entre thread y thread.

Loop Count: ☐ Forever 1

Veces que se repite el test. Forever es hasta pararlo.

☐ Delay Thread creation until needed

☐ Scheduler

### Scheduler Configuration

Duration (seconds)

Startup delay (seconds)

# Test plan

Configuration element: Establecen variables y valores default que pueden ser usados por samplers.

Listeners: Son componentes que escuchan los resultados de las pruebas. También observan, guardan y leen resultados.

Timer: Se utilizan para pausar un Thread. Representan el tiempo que un usuario tarda en realizar cierta tarea.

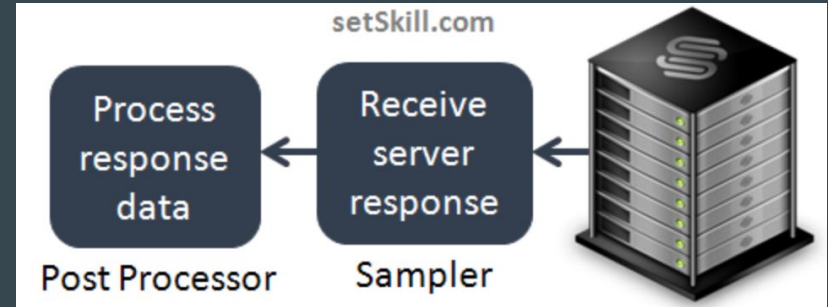
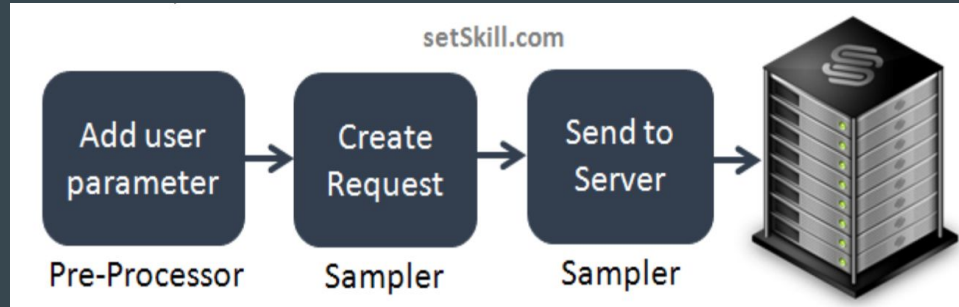
Assertions: Checkean el resultado de los samples y determinan si son exitosos o no.

Test fragment: Se utiliza para minimizar la cantidad de código en un script.

# Test plan - Pre/Post processors

Los Pre Processors se utilizan para modificar los samplers antes de que se ejecuten.

Los Post Processors se aplican luego de los samplers



# JMeter - Uso y buenas prácticas

- En las pruebas de rendimiento se debe evitar el ruido o alteraciones sobre el objetivo de prueba
- Filtrar resultados quita cargas innecesarias en las pruebas

En la documentacion de JMeter hay un manual de buenas practicas

<http://jmeter.apache.org/usermanual/best-practices.html>



# Test de estrés

- El test consiste en correr muchos threads para probar cuantos requests/seg soporta el servidor y también la latencia promedio.
- Se testeó la funcionalidad de envío de imagen (que se realiza por web API) con una imagen de poco tamaño.
- Para esto se usó un Thread Group configurado con  $n$  threads concurrentes y un Loop Controller que se encarga de lanzar los request indefinidamente para cada thread.
- Las pruebas se hicieron para  $n=$ ,  $n=$  y  $n=$ .

# Test de estrés - Resultados

## 1 usuario concurrente

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
POST Small ...	209	61	19	187	16.80	0.48%	14.3/sec	1.46	5932.66
TOTAL	209	61	19	187	16.80	0.48%	14.3/sec	1.46	5932.66

## 2 usuarios concurrentes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
POST Small ...	212	147	25	1353	171.13	3.30%	12.2/sec	9.42	5025.34
TOTAL	212	147	25	1353	171.13	3.30%	12.2/sec	9.42	5025.34

## 5 usuarios concurrentes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
POST Small ...	251	456	45	2213	419.93	11.95%	10.2/sec	29.80	4164.62
TOTAL	251	456	45	2213	419.93	11.95%	10.2/sec	29.80	4164.62

## 25 usuarios concurrentes

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/sec	Sent KB/sec
POST Small ...	372	4204	27	23664	2709.52	34.14%	5.6/sec	44.46	2181.44
TOTAL	372	4204	27	23664	2709.52	34.14%	5.6/sec	44.46	2181.44

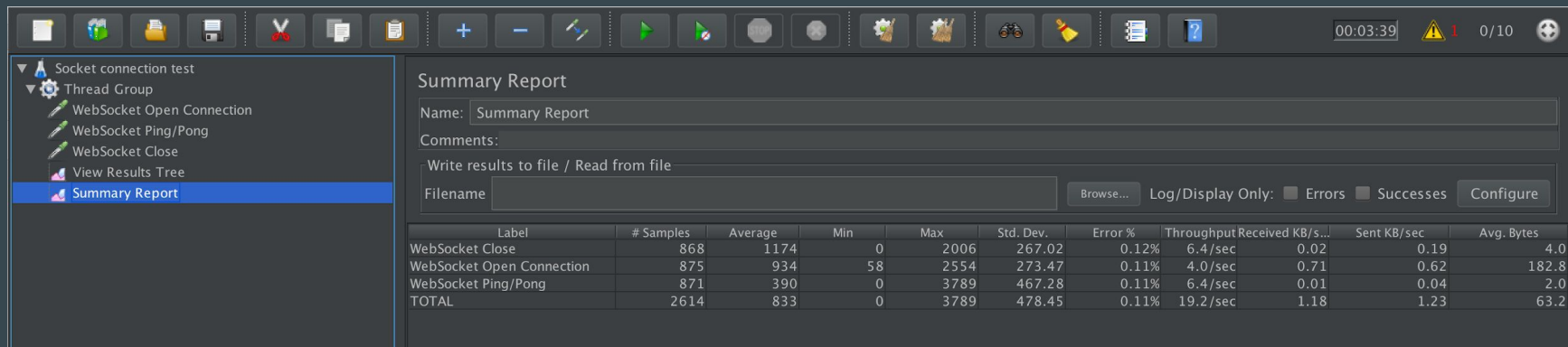
# Test distribuido

- Se puede hacer un test en clientes distribuidos que se comunican con un server.
- Para esto se corre una instancia de Jmeter en modo server en el servidor e instancias cliente de Jmeter en los clientes.
- Dado que levantar varios threads para simular clientes en el servidor es caro, este tipo de test alivia de carga al servidor y además hace más transparente el análisis de resultados.
- Mas info en <https://jmeter.apache.org/usermanual/remote-test.html>.

# Integración continua con Jmeter + Jenkins

- Jenkins puede ser utilizado como un servidor para integración continua.
- Combinado con Jmeter (más un plugin) permite poder medir los impactos de performance en el código que se coloca en un repositorio para verificar si cumple con esos atributos de calidad.
- Mas info en <https://www.blazemeter.com/blog/continuous-integration-101-how-run-jmeter-jenkins>

# Probando websocket con JMeter



The screenshot shows the JMeter Summary Report for a 'Socket connection test'. The test includes a 'Thread Group' with four sub-elements: 'WebSocket Open Connection', 'WebSocket Ping/Pong', 'WebSocket Close', and 'View Results Tree'. The 'Summary Report' is selected, displaying a table of results. The table includes columns for Label, # Samples, Average, Min, Max, Std. Dev., Error %, Throughput, Received KB/s, Sent KB/sec, and Avg. Bytes. The results show that the test was successful with a throughput of 19.2/sec and an average response time of 833ms.

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	Received KB/s...	Sent KB/sec	Avg. Bytes
WebSocket Close	868	1174	0	2006	267.02	0.12%	6.4/sec	0.02	0.19	4.0
WebSocket Open Connection	875	934	58	2554	273.47	0.11%	4.0/sec	0.71	0.62	182.8
WebSocket Ping/Pong	871	390	0	3789	467.28	0.11%	6.4/sec	0.01	0.04	2.0
TOTAL	2614	833	0	3789	478.45	0.11%	19.2/sec	1.18	1.23	63.2

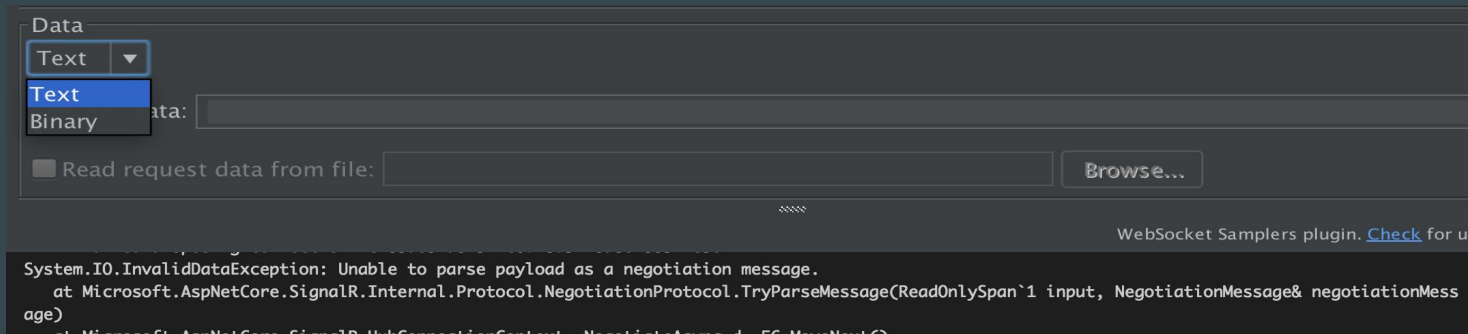
1. Samples - Número de requests enviados
2. Avg - El promedio de todas las respuestas (sum todos los tiempos / cant)
3. Tiempo de respuesta mínimo (ms)
4. Tiempo de respuesta máximo (ms)
5. Desviación estándar
6. Porcentaje de test fallidos
7. Throughput - # de request que el server tomó OK
8. Avg. Bytes - Promedio tamaño respuesta

# Dificultades - Parte 1

JMeter no está preparado para utilizarlo con SignalR y no soporta la comunicación por Web Sockets por defecto (solo a través de plugins).

Por otro lado intentamos utilizar Web Sockets utilizando JMeter y para esto probamos este plugin: <https://bitbucket.org/pjtr/jmeter-websocket-samplers/downloads/>

Problema: El plugin no envía correctamente los datos en formato json:



# Dificultades - Parte 2

- Microsoft recomienda utilizar una herramienta propia llamada “Crank” para hacer pruebas de carga en proyectos que utilicen SignalR. Pero esta herramienta solo soporta la version para ASP.NET (nosotros utilizamos Net Core).
- Muy poca documentación y/o ejemplos sobre cómo realizar este tipo de pruebas con SignalR.
- La versión de SignalR para Net Core está aún en desarrollo, no posee una comunidad grande ni una documentación exhaustiva.

# Alternativas

- Netling
- Websurge
- Scala + Gatling