

JMeter

- Propiedades emergentes no funcionales:

Hay dos tipos de propiedades emergentes, las funcionales y las no funcionales

- Categorías de propiedades no funcionales

- Fiabilidad: probabilidad de funcionamiento sin fallos durante un tiempo determinado en un entorno específico

- Disponibilidad: tiempo durante el cual el sistema ofrece servicio al usuario

- Mantenibilidad: Capacidad de un sistema para soportar cambios de los tipos: correctivos, adaptativos o perfectivos.

- Escalabilidad: hace referencia a la capacidad de mantener el tiempo de respuesta ante cambios en el número de usuarios que usan el sistema.

- Robustez: capacidad de un sistema de seguir funcionando a pesar de fallos.

Métricas: los criterios de aceptación deben incluir propiedades emergentes "científicas"

Para juzgar en qué grado se satisfacen en los criterios de aceptación se utilizan diferentes métricas:

- Fiabilidad: se utilizan pruebas aleatorias basándose en el perfil operacional. Se utilizan métricas MTTF → Mean time to failure, MTTR → mean time to repair y MTBF → Mean time between failures.

- Disponibilidad: se usa MTTR para medir el downtime del sistema

- Mantenibilidad: se utiliza el MTTR

- Escalabilidad: se cuenta la cantidad de transacciones por unidad de tiempo

Ejemplos de pruebas:

- Carga: validan el rendimiento de un sistema en términos de "tratar un número específico de usuarios manteniendo un ratio de transacciones".

- Stress: consiste en forzar peticiones al sistema por encima del límite de diseño del software, además comprueban la fiabilidad y robustez.

Resumen del proceso de pruebas no funcionales

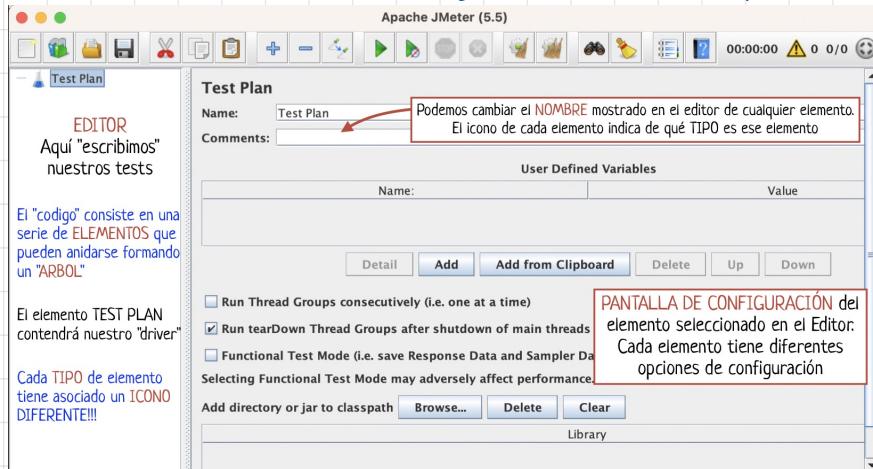
Las pruebas no funcionales determinan el rendimiento de nuestra aplicación.

Para evaluar el rendimiento necesitamos: Identificar los criterios de aceptación, Diseñar los test, Preparar el entorno de pruebas, automatizar las pruebas, Analizarnos los resultados.

- No hay que dejar las pruebas funcionales de rendimiento para final del proyecto, ya que pueden llevarnos a retocar el código.

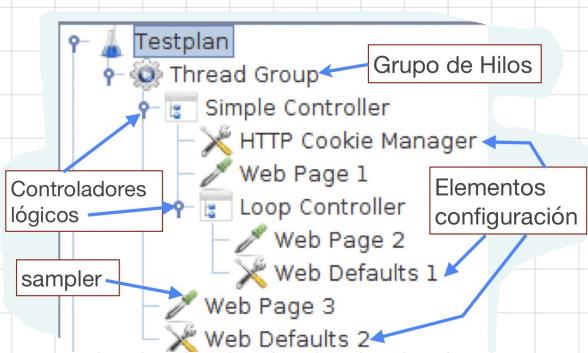
JMeter: es una herramienta 100% de escritorio Java diseñada para medir el rendimiento mediante pruebas de carga.

- Permiten múltiples hilos de ejecución concurrente, procesando diversas y diferentes patrones de petición.



Plan de pruebas: Un plan de pruebas JMeter describe una serie de pasos o acciones. Los planes de pruebas están formados por:

- Thread groups: uno o más grupos de hilos
- Logic Controllers: controladores lógicos
- Samplers, Listeners, Timers, Assertions, Pre-processors y Post Processors
- Configuration Elements



El orden de ejecución de un plan es el siguiente:

1. Configuration Elements
2. Pre-processors
3. Timers
4. Sampler
5. Post-Processors
6. Assertions
7. Listeners.

JMeter: Hilos de ejecución

Un hilo de ejecución es el punto de partida de cualquier plan de pruebas. Cada hilo ejecuta completamente el plan de forma independiente de otros hilos. Cada hilo representa un usuario.

JMeter : Samplers

Los samplers envían peticiones a un servidor. Ejemplos de samplers: HTTP request, FTP request, JDBC request, se ejecutan en el orden que aparecen en el plan.

HTTP Request

Name: HTTP Request

Comments: Esto es una petición HTTP

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: Port Number:

HTTP Request

GET Path: Content encoding:

Redirect Automatically Follow Redirects Use KeepAlive Use multipart/form-data

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	URL Encode?	Content-Type	Include Equals?
-------	-------	-------------	--------------	-----------------

Podemos configurar diferentes propiedades en un HTTP Request.
Adicionalmente, también podemos configurar algunas propiedades añadiendo uno o varios Configuration Elements en el plan.

Podemos usar muchos tipos de SAMPLERS!!!

Detail Add Add from Clipboard Delete Up Down



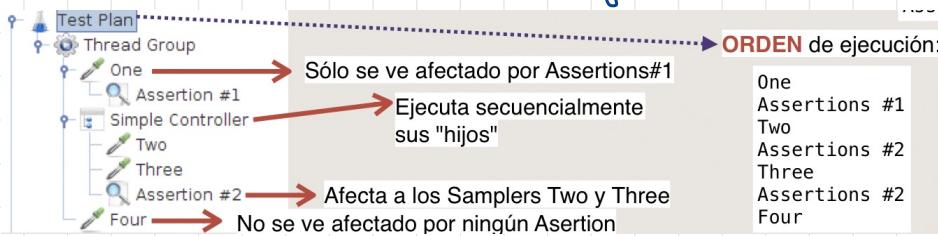
- No todos los elementos pueden anidarse dentro de otros. Un sampler puede anidarse en un grupo de hilos y/o en un controlador.

- Su ejecución puede verse afectada por otros elementos.

- En un sampler podemos anidar los siguientes elementos



- Los elementos anidados en un sampler sólo afectan a dicho sampler

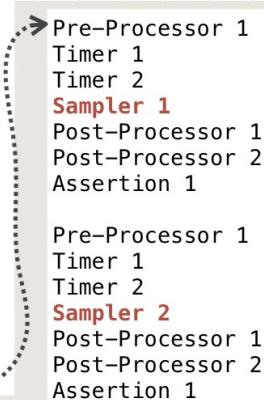


- El orden de ejecución del plan no depende del orden en el que aparecen en el árbol los diferentes elementos

Por ejemplo, en el siguiente Test Plan:

- Controller
 - Post-Processor 1
 - Sampler 1
 - Sampler 2
 - Timer 1
 - Assertion 1
 - Pre-Processor 1
 - Timer 2
 - Post-Processor 2

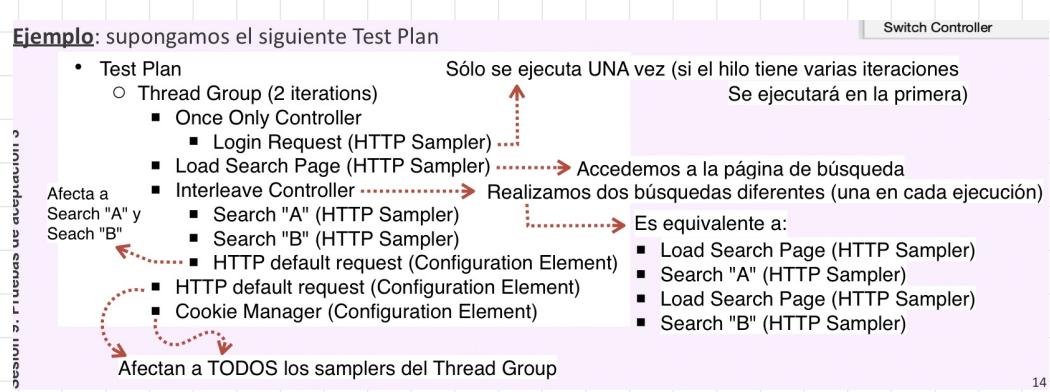
El ORDEN de ejecución es:



JMeter : Controladores lógicos : Determinan la lógica que utiliza JMeter para realizar las peticiones. Organizan el flujo de control. Actúan sobre sus elementos hijos.

- Simple Controller: No tiene efecto sobre cómo procesa JMeter los elementos hijos de dicho controlador simplemente para "agrupar" dichos elementos.
- Loop Controller: Itera sobre los elementos hijos un cierto número de veces.
- Only Once Controller: Los elementos hijos solo tienen que ser ejecutados una vez por hilo de ejecución.
- Interleave Controller: ejecutará uno de esos subcontroladores o samplers en cada iteración del bucle de pruebas, alterándose secuencialmente a lo largo de la lista.

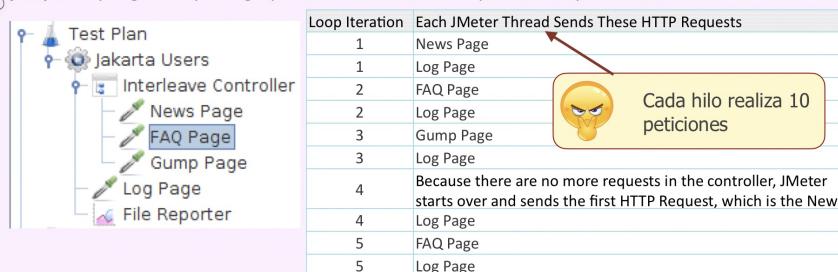
Ejemplo: supongamos el siguiente Test Plan



14

Otro ejemplo:

Ejemplo: supongamos que el grupo de hilos está formado por 2 hilos, y 5 iteraciones



- El controlador Interleave Controller sólo "controla" sus samplers hijos (es decir: "News Page", "FAQ Page", y "Gump Page")
- El sampler "Log Page" se ejecutará siempre, independiente de la iteración de cada hilo
- El Listener "File Reporter" recopila los resultados de la ejecución de todos los elementos del grupo de hilos ("afecta" a todos los elementos de su mismo nivel y siempre se ejecuta en último lugar, aunque en el árbol aparezca "escrito" antes!!!)

JMeter : Elementos de configuración

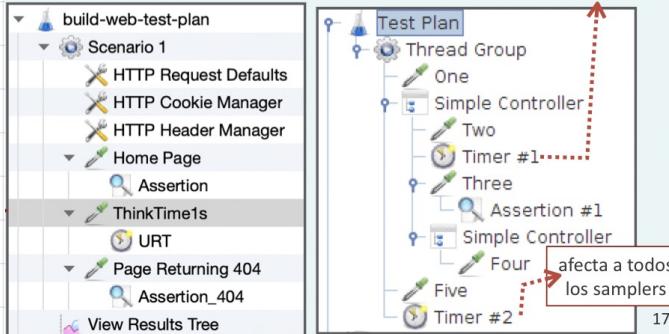
- Trabajan conjuntamente con los samplers, modificando las peticiones
- Un elemento de configuración es accesible sólo dentro de la rama del árbol en la que se sitúa el elemento
- Un elemento de configuración dentro de una rama del árbol tiene mayor preferencia que el mismo elemento en una rama padre.
- HTTP Request default para cuando varios samplers comparten URL o parte de ella.
- HTTP Cookies Manager permite almacenar y enviar cookies como si fuese un navegador.



Timers por defecto JMeter realiza las peticiones sin hacer ninguna pausa, lo que podría saturar el servidor. Los temporizadores permiten introducir pausas antes de las peticiones de cada hilo.

- Constant Timer: Retrasa cada petición de usuario la misma cantidad de tiempo
- Uniform Random Timer: Introduce pausas aleatorias
- Gaussian Random Timer: introduce pausas según la distribución gaussiana.
- Si hay varios timers en el mismo nivel del sampler se aplicarán todos ellos.
- Si queremos que un timer solo afecte a un sampler hay que añadirlo como su hijo
- Si queremos que un timer se aplique después de un sampler, lo añadiremos al siguiente sampler o lo añadiremos como hijo de un Flow Control Action Sampler

Es de un sampler:



Aserciones Las aserciones permiten hacer afirmaciones sobre las respuestas recibidas del servidor que se está probando. Podemos añadir aserciones de cualquier sampler. Se trata de probar que la aplicación devuelva el resultado esperado.

Test Plan

Thread Group

HTTP Request JMeter Home

Response Assertion

Assertion Results

WorkBench

Response Assertion

Name: Response Assertion

Comments:

Apply to:

Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use

Field to Test

Text Response Response Code Response Message Response Headers

Request Headers URL Sampled Document (text) Ignore Status

Request Data

Pattern Matching Rules

Contains Matches Equals Substring Not Or

Patterns to Test

Add Add from Clipboard Delete

Cualquier driver SIEMPRE debe incluir aserciones!!!

Listeners se utilizan para ver o guardar en el disco las peticiones realizadas. Proporcionan acceso a la información que JMeter va acumulando sobre los casos de prueba a medida que se ejecutan los test. Todos los listeners guardan los mismos datos, solo cambian lo que representan.

Simple Data Writer

Name: Simple Data Writer

Comments:

Write results to file / Read from file

Filename Log/Display Only: Errors Successes

Aggregate Report

Name: Aggregate Report

Comments:

Write results to file / Read from file

Filename

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	Received KB/sec	Sent KB/sec
HTTP Request-3	80	222	220	312	104	353	0.0%	1.0/sec	2.62	0.45
HTTP Request-1	80	231	233	336	108	354	0.0%	1.5/sec	10.49	0.33
HTTP Request-2	80	233	233	324	102	354	1.25%	1.5/sec	5.23	0.28
TOTAL	240	229	229	330	102	354	0.42%	4.4/sec	18.18	0.86

Aggregate Graph

Name: Aggregate Graph

Comments:

Write results to file / Read from file

Filename

Label	# Samples	Average	Median	90% Line
A,B,C,D,E	4,414	4,302	4,073	
A,B,C,D,E,F	4,119	4,371	4,560	
A,B,C,D,E,F,G	4,734	4,737	5,779	
A,B,C,D,E,F,G,H	3,208	3,227	3,754	
A,B,C,D,E,F,G,H,I	3,355	3,312	4,050	
A,B,C,D,E,F,G,H,I,J	3,355	3,312	4,050	

Graph Results

Name: Graph Results

Comments:

Write results to file / Read from file

Filename

Graphs to Display: Data Average Median Deviation Throughput

620 ms

0 ms

No of Samples 832

Latest Sample 203

Average 298

Deviation 216

Median 204

El Throughput representa la capacidad del servidor para soportar una determinada carga. Cuanto más alto sea, mayor será el rendimiento del servidor!!!

Los samplers calculan:

- #Samples: Número de muestras con la misma etiqueta.
- Average: Tiempo medio de las respuestas
- Mediana
- Línea de 90%
- Min
- Max
- % Error
- Kb/sec: rendimiento expresado en KB/sec

Vamos a sacar el plan de la siguiente salida

Home
Main Catalog
Login
Reptiles
Sign Out
Home
Main Catalog
Login
Birds
Sign out
Home
Main
Main Catalog
Login
Dogs
Sign Out

