

PtU Certified Performance Tester con JMeter (CPTJM)
Programa de estudios

Liberado
Versión 1.0 2019

Performance Testing United



Derechos de autor

Este documento puede ser copiado en su totalidad, o se pueden hacer extractos, si se reconoce la fuente.

Todos los recursos de Performance Testing United, incluyendo este documento son © Performance Testing United (en adelante denominado PtU).

Los autores del material y los expertos internacionales involucrados en la creación de los recursos de PtU transfieren los derechos de autor a Performance Testing United (PtU). Los autores del material, los expertos internacionales y PtU han acordado las siguientes condiciones de uso:

- Cualquier persona o empresa de formación puede utilizar este plan de estudios como base para un curso de formación, si PtU es reconocido como fuente y propietario de los derechos de autor del plan de estudios, una vez que hayan sido reconocidos oficialmente por PtU. Más información sobre el reconocimiento está disponible en: <https://www.pt-united.com/recognition>
- Cualquier persona o grupo de personas puede usar este programa de estudios como base para artículos, libros u otros escritos derivados si PtU es reconocido como fuente y propietario de los derechos de autor del programa de estudios.

Gracias a los principales colaboradores

Delvis Echeverria, Guillermo Skrilec, Rahul Verma.

Gracias al comité de revisión

Alexis Herrera, Alfonso Fernández, Almidena Vivanco, Ángel Rayo Acevedo, Arjan Brands, Bruno, Mareque Acosta, Daniel Tolosa, Eduardo Ricardo Delgado Cortés, Emmanuel Espinoza Sales, Guino Henostroza, Gustavo Marquez Sosa, Gustavo Terrera, Héctor Revalcaba, Jefferson Pereira Ortiz, Juan Pablo Rios Alvarez, Leandro Melendez, Manuel Fischer, Marcella Mellado, Marelis V. Pérez García, Michael Frontner, Miguel Angel De León Trejo, Nadia Soledad Cavalleri, Rafael Márquez Galicia, Richard Seidl & Silvia Nane.

Historial de revisiones

Versión	Fecha	Observaciones
PtU 2019	Septiembre 2019	Primer lanzamiento oficial

Tabla de contenidos

Propósito de este documento	5
Recursos de PtU	5
¿Qué son las pruebas de rendimiento?	5
Sobre PtU Certified Performance Tester con JMeter (CPTJM)	5
Objetivos de negocio	6
Objetivos de aprendizaje/niveles cognitivos de conocimiento	6
Requisitos previos	8
Capítulo 1 - Introducción	9
1.1 Principales conceptos sobre pruebas de rendimiento	9
1.2 Conceptos básicos de JMeter	17
Capítulo 2 - Scripts básicos	20
2.1 Plan de pruebas	20
2.2 Grabación y ejecución	26
2.3 Receptores de resultados	28
Capítulo 3 - Scripts avanzados	30
3.1 Correlación	30
3.2 Parametrización	32
3.3 Tiempos de espera	34
3.4 Controladores lógicos	35
3.5 Aserciones	36
3.6 Depuración del script	38
Capítulo 4 - Ejecución de pruebas	39
4.1 Ejecución de las pruebas	39
4.2 Monitoreo de la aplicación	41
Capítulo 5 - Documentación	44
5.1 Introducción	44
5.2 Plan de pruebas de rendimiento	44
5.3 Guión de pruebas	45
5.4 Informe de resultados	45

Capítulo 6 - Extra	46
6.1 Mejores prácticas de JMeter	46
6.2 Pruebas de servicios web	46
Glosario	47
Referencias	48

Propósito de este documento

Este temario constituye la base de Performance Testing United en lo que respecta a su certificación Performance Tester con JMeter (CPTJM). Este documento define lo que usted necesita saber para aprobar la prueba de certificación Performance Tester con JMeter (CPTJM). El examen de certificación sólo cubrirá los conceptos y conocimientos que se describen en este documento.

Recursos de PtU

Una visión general de los recursos de PtU, así como toda la información relevante sobre la certificación PtU y otros tipos de certificaciones PtU están disponibles en www.pt-united.com — el sitio oficial de Performance Testing United. La información en www.pt-united.com incluye:

- Una lista completa de los proveedores de capacitación reconocidos y de los cursos disponibles. Tenga en cuenta que se recomienda la capacitación, pero no es necesaria para realizar el examen de certificación PtU CPTJM.
- El programa de estudios para descargar (este documento).
- Una muestra completa de un conjunto de 40 preguntas de PtU CPTJM, con sus respuestas para fines de entrenamiento.
- Nuestro objetivo es que los documentos estén disponibles en otros idiomas lo antes posible. Para ver las versiones en los idiomas actualmente disponibles, por favor visite www.pt-united.com.

¿Qué son las pruebas de rendimiento?

Las pruebas de rendimiento son un tipo específico de pruebas que permiten determinar la capacidad de respuesta de un sistema frente a una carga de trabajo en un contexto particular. Asimismo permite verificar y validar atributos de calidad del software como la escalabilidad, fiabilidad, disponibilidad, uso de recursos, entre otros aspectos no funcionales.

Existen distintas herramientas que pueden ser utilizadas para ejecutar pruebas de rendimiento, JMeter es una de las herramientas más utilizadas a nivel mundial, es gratuita y de código abierto.

Sobre PtU Certified Performance Tester con JMeter (CPTJM)

PtU es un curso práctico para testers que ya se encuentran involucrados en pruebas de rendimiento, o para aquellos que desean comenzar en esta área. El curso se encuentra enfocado en el uso de la herramienta JMeter, para llevar a cabo pruebas de rendimiento en aplicaciones web, acercando de esta forma el uso de la herramienta a implementaciones reales a través de conceptos como: grabación de las pruebas, construcción de scripts que permitan simular el uso real del sistema, ejecución de las pruebas, monitoreo de la aplicación durante las

pruebas, entre otros conceptos. PtU tiene una sólida base técnica apropiada para testers de software, ingenieros de prueba, analistas de performance, líderes de pruebas, administradores de calidad y soporte a la operación.

Objetivos de negocio

(BOs por sus siglas en inglés “Business Objectives”):

BO1	Comprender los principales conceptos de las pruebas de rendimiento y la metodología para llevarlas a cabo.
BO2	Utilizar la herramienta JMeter para crear y ejecutar pruebas de rendimiento en aplicaciones web.
BO3	Analizar los resultados de las pruebas de rendimiento e identificar mejoras del rendimiento de la aplicación.
BO4	Identificar indicadores de rendimiento de una aplicación y utilizar herramientas para el monitoreo.

Objetivos de aprendizaje/niveles cognitivos de conocimiento

Los objetivos de aprendizaje (LOs) son breves declaraciones que describen lo que se espera que usted sepa después de estudiar cada capítulo. Los objetivos de aprendizaje se definen de acuerdo a los siguientes niveles:

- K1: Recordar
- K2: Comprender
- K3: Aplicar

La siguiente tabla enumera los principales LOs para la certificación PtU-CPTJM (LOs por sus siglas en inglés “Learning Objectives”):

LO1	Comprender que son las pruebas de rendimiento y sus diferentes tipos. (K2)
LO2	Comprender la metodología para las pruebas de rendimiento. (K2)
LO3	Identificar los distintos tipos de herramientas utilizadas para las pruebas de rendimiento. (K1)
LO4	Comprender los aspectos básicos del protocolo HTTP(S). (K2)

LO5	Comprender por qué se utiliza JMeter para las pruebas de rendimiento. (K2)
LO6	Instalar y ejecutar JMeter en Windows y Linux. (K3)
LO7	Comprender y aplicar los principales elementos para construir un plan de pruebas en JMeter. (K2)
LO8	Crear scripts básicos grabando una sesión y ejecutando la misma en JMeter. (K3)
LO9	Analizar los resultados de las pruebas a través de distintos reportes. (K3)
LO10	Comprender el concepto de correlación y cómo utilizar expresiones regulares. (K2)
LO11	Aplicar el uso de expresiones regulares para manejar la correlación en JMeter. (K3)
LO12	Comprender el concepto de parametrización. (K2)
LO13	Construir y configurar fuentes de datos para utilizar en el script de pruebas. (K3)
LO14	Comprender y aplicar temporizadores, aserciones y controladores en JMeter. (K3)
LO15	Depuración de un script en JMeter. (K3)
LO16	Preparación de los scripts para la ejecución de las pruebas. (K3)
LO17	Ejecución de las pruebas utilizando el modo línea de comandos. (K3)
LO18	Ejecución de las pruebas a través del modo distribuido. (K1)

LO19	Comprender cómo se realiza el monitoreo de los recursos del sistema durante las pruebas de rendimiento y sus principales indicadores. (K2)
LO20	Aplicar herramientas básicas de monitoreo durante la ejecución de las pruebas de rendimiento. (K3)
LO21	Documentación sobre las pruebas de rendimiento. (K2)
LO22	Comprender las mejores prácticas a la hora de utilizar JMeter. (K2)
LO23	Ejecución de scripts para servicios web. (K3)

Requisitos previos

- Conocimientos básicos de programación. Conocer los conceptos de variable, función y estructuras de control (condicionales y bucles).
- Conocimientos básicos de aplicaciones web y su arquitectura. Conocer la arquitectura cliente-servidor.
- Conocimientos básicos del protocolo HTTP(S). Conocer los conceptos de petición, respuesta, cookies, parámetros de la URL, métodos de invocación (GET/POST), encabezados del mensaje y cuerpo del mensaje.

Capítulo 1 - Introducción

LO1	Comprender que son las pruebas de rendimiento y sus diferentes tipos. (K2)
LO2	Comprender la metodología para las pruebas de rendimiento. (K2)
LO3	Identificar los distintos tipos de herramientas utilizadas para las pruebas de rendimiento. (K1)
LO4	Comprender los aspectos básicos del protocolo HTTP(S). (K2)
LO5	Comprender por qué se utiliza JMeter para las pruebas de rendimiento. (K2)
LO6	Instalar y ejecutar JMeter en Windows y Linux. (K3)

1.1 Principales conceptos sobre pruebas de rendimiento

1.1.1 Introducción a las pruebas de rendimiento

Las pruebas de rendimiento permiten estudiar el desempeño de un sistema cuando el mismo se enfrenta a escenarios de carga y estrés, similares a los que pueden suceder en producción.

Estas pruebas permiten conocer, entre otros aspectos, la cantidad de usuarios simultáneos que soporta el sistema, obtener datos para el dimensionamiento de la infraestructura necesaria para un sistema, así como brindar información para mejorar los tiempos de respuesta del sistema.

Para llevar a cabo las pruebas se deben definir previamente los escenarios de carga que serán simulados a través de scripts automatizados, que representan la operativa de los usuarios en el sistema.

Utilizando herramientas de generación de carga, se ejecutan los scripts definidos y se realiza el monitoreo de los distintos aspectos de la aplicación, como los tiempos de respuesta, consumo de recursos, entre otros.

Como resultado, se podrán detectar problemas de configuración del hardware y software, problemas en los enlaces, incluso problemas de rendimiento debido a la implementación realizada.

Las pruebas de rendimiento son el mecanismo más efectivo para minimizar los riesgos en la puesta en producción, como por ejemplo tiempos de respuesta elevados, consumo inadecuado de recursos, o interrupciones en el servicio.

1.1.2 Tipos de pruebas de rendimiento

Existen distintos tipos de pruebas de rendimiento (performance), donde cada uno de ellos persigue distintos objetivos.

- Pruebas de carga (load)
- Pruebas de estrés (stress)
- Pruebas de pico (spike)
- Pruebas de resistencia (endurance)
- Pruebas de escalabilidad (scalability)

1.1.2.1 Pruebas de carga

Las pruebas de carga se realizan para evaluar el comportamiento del sistema, bajo una cantidad esperada de usuarios concurrentes, que realizan un número específico de transacciones durante un tiempo predefinido.

Estas pruebas permiten medir los tiempos de respuesta de las transacciones, así como evaluar si los recursos del sistema limitan el desempeño de la aplicación.

1.1.2.2 Pruebas de estrés

Las pruebas de estrés se realizan para determinar el comportamiento del sistema cuando se enfrenta a cargas extremas. Se realizan aumentando el número de usuarios concurrentes y el número de transacciones que ejecutan, sobrepasando la carga esperada.

Permiten conocer cómo es el rendimiento del sistema, en caso de que la carga real supere la carga esperada, determinando cuáles son los componentes y/o recursos que fallan primero y limitan el desempeño del sistema.

1.1.2.3 Pruebas de pico

Este tipo de pruebas son muy similares a las pruebas de estrés, pero se realizan en periodos cortos de tiempo, simulando importantes cambios de carga en un momento dado.

Permiten conocer el comportamiento del sistema cuando existe un pico de uso del mismo, y evaluar si luego de este el sistema es capaz de retornar a un estado estable.

1.1.2.4 Pruebas de resistencia

Las pruebas de resistencia permiten determinar si el sistema puede soportar una carga esperada de forma continua, durante un periodo de tiempo acorde al contexto de uso del sistema.

Permiten evaluar el desempeño de los distintos recursos, por ejemplo si existen fugas de memoria, degradaciones por mal manejo de las conexiones a la base de datos, entre otros.

1.1.2.5 Pruebas de escalabilidad

Las pruebas de escalabilidad se realizan para evaluar la capacidad de crecimiento del sistema. Típicamente se proyecta a futuro, por ejemplo, la cantidad de usuarios concurrentes, el número de transacciones que pueden realizar, el volumen de información en la base de datos, entre otros aspectos no funcionales del sistema.

1.1.3 Metodología de pruebas de rendimiento

La metodología para llevar a cabo las pruebas de rendimiento es la siguiente:

1.1.3.1 Planificación de las pruebas

Esta fase consiste en el armado del plan de pruebas de rendimiento. En el mismo se definen los escenarios de prueba, donde se identifican los distintos tipos de usuarios que interactúan con el sistema y los flujos que realizan. Se determina la cantidad de usuarios en cada escenario y la cantidad de transacciones que ejecutarán para simular el uso del sistema.

Como parte del plan de pruebas de rendimiento, también se debe tener en cuenta:

1.1.3.1.1 Datos de prueba

En este punto se definen tanto los datos de la base de datos, como los datos que se usarán como entrada para las transacciones a ejecutar. Es necesario que los datos sean similares a los que se espera tener en producción, tanto en calidad como en cantidad; de no ser así, no se estaría modelando adecuadamente la realidad y muchos de los problemas potenciales podrían no llegar a observarse en las pruebas.

1.1.3.1.2 Infraestructura de pruebas

En este punto se define la infraestructura sobre la cual se ejecutarán las pruebas. En este sentido, existen distintas opciones, a continuación se presentan dos alternativas:

- La primera, es ejecutar en la infraestructura de producción, ya sea porque no se está usando o porque es posible definir una ventana de tiempo para utilizarla.
- De no ser posible, se deberá armar una infraestructura igual a la de producción para llevar a cabo el proceso de pruebas.

El ambiente de pruebas deberá ser monitoreado. Se definen indicadores primarios que permiten observar el comportamiento de los distintos recursos, y, frente a la necesidad de obtener más información, se podrán implementar indicadores de segundo nivel.

1.1.3.1.3 Criterio de aceptación

Es importante establecer uno o más criterios de aceptación u objetivos a alcanzar. Este puede ser definido en base a una cantidad de operaciones a ejecutar en una ventana de tiempo, al consumo de recursos del sistema o al tiempo de respuesta de cada transacción, cualquiera de ellos medido en la situación de carga del sistema definida por el escenario correspondiente. Los criterios de aceptación deben ser definidos de forma temprana, y es una buena práctica que los mismos sean parte de los requerimientos del sistema. Cuando se cumplen los criterios de aceptación, las pruebas de rendimiento se dan por finalizadas.

1.1.3.2 Especificación de escenarios de prueba

Se define en detalle las pruebas que componen cada uno de los escenarios. Se utiliza una planilla donde se detalla cada paso y la verificación final que se debe realizar para asegurar que la prueba haya concluido de forma exitosa. Estos

escenarios son validados con integrantes del equipo técnico y del equipo de negocio, para asegurar que se haya modelado el comportamiento que se espera en la realidad.

1.1.3.2.1 Escenarios de prueba

Los escenarios especifican las diferentes condiciones de uso que debe soportar el sistema cuando se utilice en producción. Generalmente, se asocia un escenario a un determinado momento del día. Por ejemplo, se puede definir un “escenario diurno” como la operativa que determina la carga que sufre el sistema a diario de 1:00 PM a 2:00 PM. Esa carga prevista se toma como referencia y se le denomina escenario del 100%, luego se ejecutan distintos porcentajes de carga a través de diferentes transacciones.

Además, se debe especificar la cantidad de usuarios que ejecuta cada transacción, la cantidad de iteraciones de cada usuario, la forma de ingreso al sistema por parte de los usuarios y los procesos que pueden ejecutar en paralelo en dicho escenario.

1.1.3.2.2 Guiones de prueba

Los guiones de pruebas son el análogo de los casos de prueba, para las pruebas de rendimiento. Se debe definir un guión de pruebas preciso para cada transacción, donde se incluye cada acción que realiza el usuario en el sistema, como es realizada la misma y la correspondiente respuesta del sistema.

1.1.3.3 Automatización de pruebas

La etapa de automatización tiene como objetivo construir los scripts que llevarán a cabo las transacciones, siguiendo el guión definido en la etapa anterior.

Para la construcción de los scripts, existen varias herramientas que permiten la creación de los mismos, y luego la ejecución de manera eficiente de cientos de usuarios virtuales. Estas herramientas proveen la capacidad de ingresar los distintos parámetros del escenario definido, como la cantidad de usuarios que ejecuta cada transacción, la cantidad de veces que ejecuta cada usuario, la forma de ingreso de los usuarios al sistema (cadencia), el tiempo en el que se ejecuta el escenario, entre otros.

Esta etapa de la metodología es el foco de Performance Testing United, y la herramienta a utilizar es Apache JMeter.

1.1.3.4 Armado del ambiente

El armado del ambiente de pruebas consiste en distintas actividades que permiten la preparación del ambiente de ejecución.

Entre las principales actividades se encuentran:

- Preparación y configuración de las máquinas generadoras de carga.
- Instalación de las herramientas de monitoreo en los distintos componentes del sistema, y configuración de los indicadores seleccionados.
- Preparación del ambiente donde se encuentra desplegado el sistema, con sus datos y configuraciones necesarias.
- Implementación de decisiones relacionadas a la interoperabilidad del sistema.

1.1.3.5 Ejecución de pruebas

Se ejecutan distintas pruebas (funcionalidades individuales, línea base, escenarios de carga) en las cuales se analiza el comportamiento del sistema, el uso de los recursos y los tiempos de respuesta obtenidos.

1.1.3.5.1 Línea base

La ejecución de la línea base consiste en ejecutar cada transacción con un único usuario y con toda la infraestructura dedicada al mismo. El objetivo es conocer los mejores tiempos que se pueden obtener en cada transacción y, de esta manera, tener un punto de referencia para analizar el impacto de los distintos escenarios de concurrencia.

1.1.3.5.2 Ejecución de escenarios

La ejecución de los escenarios definidos se realiza de manera gradual. Se debe definir cómo se incrementará la carga de acuerdo a la realidad del proyecto, aunque generalmente se aplica un incremento del 20% en cada prueba. Por este motivo se ejecuta el 20% de la carga, luego el 40%, el 60%, el 80% hasta llegar al 100% que es el escenario objetivo.

1.1.3.5.3 Monitoreo del sistema

En el transcurso de la ejecución de las pruebas, se debe realizar el monitoreo de los indicadores de desempeño del sistema, utilizando las herramientas correspondientes.

Toda la información recopilada en esta etapa deberá ser guardada para su posterior análisis.

1.1.3.6 Análisis de resultados

En base a los datos obtenidos en la etapa anterior, se deberá realizar el análisis correspondiente del desempeño del sistema y de los componentes de su arquitectura durante las pruebas.

Se sugiere apoyar el análisis utilizando herramientas gráficas que faciliten la interpretación de los datos.

1.1.3.6.1 Re-ejecución de las pruebas

Del monitoreo de los indicadores y del análisis de los resultados que se obtienen en la ejecución de los escenarios de prueba, se espera que surjan oportunidades de mejora. Estas oportunidades de mejora son aplicadas en el sistema a través de ajustes.

Luego de aplicarse los ajustes, se deben volver a ejecutar las pruebas de rendimiento para evaluar el impacto de los ajustes realizados.

1.1.3.6.2 Informe de cierre de pruebas

Una vez alcanzados los objetivos, se procede a la creación de un informe de cierre de pruebas, donde se detallan los resultados obtenidos.

1.1.4 Herramientas para pruebas de rendimiento

Para llevar adelante las pruebas de rendimiento, existen distintos tipos de herramientas que se detallan a continuación.

1.1.4.1 Construcción de scripts

La construcción de los scripts se realiza utilizando la herramienta JMeter.

Dicha herramienta tiene la posibilidad de grabar un flujo en el sistema, y generar un script que se utiliza como punto de partida para las pruebas.

1.1.4.2 Generación de carga

Para la ejecución de las pruebas simulando una carga de usuarios concurrentes, se utilizan herramientas de generación de carga que utilizan como entrada el script definido. JMeter cuenta con esta capacidad y por lo tanto será utilizada también con este fin.

1.1.4.3 Monitoreo de recursos

Durante la ejecución de las pruebas, se realiza el monitoreo de los componentes del sistema. Para esto, se utilizan herramientas de monitoreo que se encargan de tomar información de distintos indicadores de los componentes a nivel de hardware y software.

1.1.5 Introducción al protocolo HTTP(S)

Miles de millones de imágenes JPEG, páginas HTML, archivos de texto, video, audio, y más, navegan a través de Internet todos los días haciendo uso del protocolo HTTP(S). El protocolo de transferencia de hipertexto, en inglés “Hypertext Transfer Protocol” (HTTP), o su versión con seguridad integrada “Hypertext Transfer Protocol Secure” (HTTPS), son los protocolos de aplicación más ampliamente usados en Internet actualmente.

Antes de comenzar, con la creación de los scripts, es importante un entendimiento profundo del protocolo HTTP(S). Entendiendo este protocolo, será posible asimilar más rápido cómo funciona la creación de los scripts en las pruebas de rendimiento.

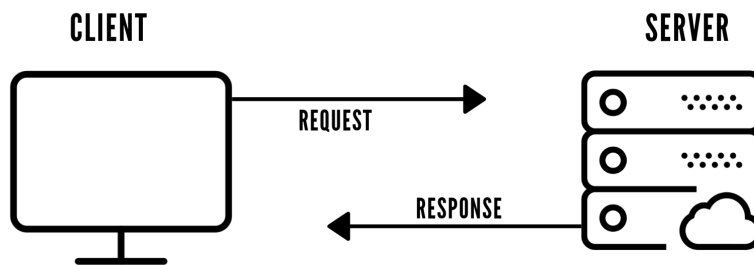
El protocolo permite la comunicación entre un cliente y un servidor. El cliente inicia una petición que es enviada al servidor y este devuelve una respuesta asociada a la petición enviada desde el cliente.

Como la mayoría de los protocolos de Internet, este es un protocolo basado en texto de peticiones y respuestas, que utiliza un modelo de comunicaciones cliente-servidor.

El protocolo HTTP(S) es un protocolo sin estado, lo que significa que el servidor no está obligado a almacenar información de la sesión y que cada solicitud es independiente de la otra.

Cuando un script es grabado usando la herramienta JMeter, la herramienta captura toda la comunicación entre el explorador y el servidor web. La información capturada se convierte en parte del script.

Cuando el script sea ejecutado a través de JMeter, se lleva a cabo una interacción con la aplicación web, simulando el comportamiento de un usuario real.



[Traducción: En la imagen se muestra del lado izquierdo el usuario (client) que interactúa a través de una petición con el servidor (server). El mismo devuelve una respuesta (response) a la petición realizada]

1.2 Conceptos básicos de JMeter

1.2.1 Introducción a JMeter

JMeter es una herramienta libre y de código abierto creada por Apache Software Foundation para llevar a cabo pruebas de rendimiento. La herramienta es una aplicación basada en Java y fue diseñada originalmente para pruebas de aplicaciones web.

Permite realizar simulaciones de carga a través de distintos tipos de aplicaciones y protocolos:

- Web - HTTP, HTTPS (Java, NodeJS, PHP, ASP.NET, etc.)
- Servicios web SOAP/REST
- FTP, FTPS
- Base de datos vía JDBC
- LDAP
- Middleware orientado a mensajes (MOM) vía JMS
- Mail - SMTP, POP3 e IMAP
- Comandos nativos o scripts shell

- TCP
- Objetos Java

JMeter cuenta con un entorno de desarrollo integrado (IDE), que permite crear rápidamente pruebas a través de su función de grabación de scripts desde un explorador web y la ejecución de las mismas.

Cuenta con un modo de ejecución a través de una línea de comando, lo que hace posible ejecutar las pruebas desde cualquier sistema operativo compatible con Java (Windows, Linux, Mac OS, entre otros).

Permite exportar los resultados de las pruebas a través de un reporte HTML completo.

JMeter permite la extensión de sus funcionalidades a través de complementos. Varios de ellos se encuentran disponibles a través de la comunidad Apache, pero también es posible construir nuevos cuando sea necesario.

Permite la integración con herramientas y librerías en esquemas de integración continua, como por ejemplo Maven, Gradle y Jenkins.

Es importante entender que JMeter no es un explorador web, la herramienta trabaja a nivel de protocolo. En particular, JMeter no ejecuta código embebido en las páginas como lo haría un explorador web, por lo tanto no ejecuta JavaScript y tampoco realiza el dibujado (render) de las páginas HTML.

1.2.2 Instalación de JMeter

Como requisito para ejecutar JMeter, es necesario tener instalado Java 8 o superior.

Para comenzar a utilizar JMeter, se descarga de la siguiente dirección: http://jmeter.apache.org/download_jmeter.cgi

JMeter se descarga en un archivo comprimido, y deberá ser descomprimido para poder ejecutarse.

Para ejecutar JMeter se debe ejecutar el archivo correspondiente al sistema operativo en el directorio “bin”. El archivo “jmeter.bat” en Windows y “jmeter.sh” en Linux y Mac OS.

JMeter se ejecutará en modo “GUI”, esto quiere decir que mostrará su interfaz gráfica para crear los scripts de prueba.

Capítulo 2 - Scripts básicos

LO7	Comprender y aplicar los principales elementos para construir un plan de pruebas en JMeter. (K2)
LO8	Crear scripts básicos grabando una sesión y ejecutando la misma en JMeter. (K3)
LO9	Analizar los resultados de las pruebas a través de distintos reportes. (K3)

2.1 Plan de pruebas

2.1.1 Principales conceptos del plan de pruebas

Un plan de prueba en JMeter define una estructura basada en árboles de cómo, cuándo y qué probar, proporcionando la ejecución de una secuencia de acciones.

Con la estructura de árbol que maneja JMeter, es importante entender que los elementos tienen una jerarquía, de manera que los de más arriba aplican a los elementos que se encuentran debajo.

Estos elementos se encuentran agrupados en categorías, lo que hace más sencillo encontrar cada uno de ellos a la hora de crear un script de prueba.

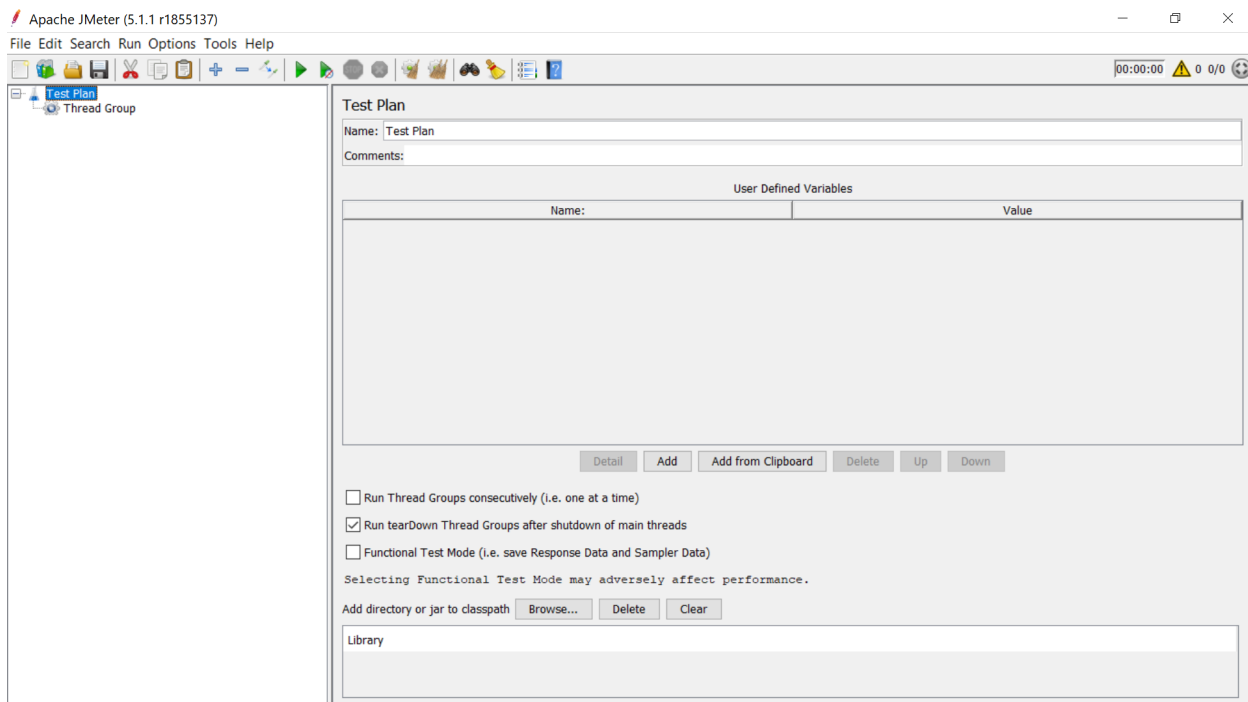
Las categorías de los elementos que maneja JMeter son:

- Peticiones (Sampler): Aquí se encuentran los distintos tipos de peticiones para diferentes protocolos.
- Controladores lógicos (Logic Controllers): Aquí se encuentran los controladores lógicos que permiten definir el flujo de las pruebas.
- Pre-procesadores (Pre Processors): Se agrupan en esta categoría todos los elementos que se pueden ejecutar antes de realizar una petición.
- Post-procesadores (Post Processors): Se agrupan en esta categoría todos los elementos que se pueden ejecutar después de realizar una petición.
- Aserciones (Assertions): Aquí se agrupan todos los elementos utilizados para realizar verificaciones y validaciones durante la ejecución de las

pruebas.

- Temporizadores (Timer): Aquí se encuentran los elementos relacionados a los tiempos de espera.
- Fragmentos de script (Test Fragment): Esta categoría contiene un solo elemento con el mismo nombre. Este elemento permite definir fragmentos del script para que sean reutilizados en distintas secciones del mismo.
- Elementos de configuración (Config Element): Dentro de esta categoría están todos los elementos de configuración para el script.
- Receptores (Listener): Aquí se encuentran los elementos relacionados a los receptores de resultados y reportes de ejecución de las pruebas.

Un plan de prueba puede incluir elementos de las categorías mencionadas anteriormente, y los mismos se ubican en el árbol que se puede ver a la izquierda de la siguiente imagen. De acuerdo al comportamiento deseado, los elementos pueden anidarse o ubicarse en paralelo con otros.



2.1.2 Hilos/Usuarios

Los hilos o usuarios se encuentran representados en JMeter a través de grupos de hilos y son el elemento inicial para comenzar a construir el plan de pruebas.

Cada grupo de hilos representa un conjunto de usuarios del sistema, y se utilizan para ejecutar las pruebas de rendimiento simulando la concurrencia de usuarios sobre la aplicación.

De los grupos de hilos dependen otros elementos, como son las peticiones HTTP(S), controladores, entre otros. Esto será presentado más adelante.

El grupo de hilos en JMeter se visualiza de la siguiente forma:

The screenshot shows the 'Thread Group' configuration window in JMeter. It includes fields for 'Name' (Thread Group), 'Comments', and 'Action to be taken after a Sampler error' with radio buttons for 'Continue', 'Start Next Thread Loop', 'Stop Thread', 'Stop Test', and 'Stop Test Now'. The 'Thread Properties' section contains 'Number of Threads (users): 1', 'Ramp-Up Period (in seconds): 1', 'Loop Count: 1' (with 'Forever' unchecked), 'Delay Thread creation until needed' (unchecked), and 'Scheduler' (unchecked). The 'Scheduler Configuration' section has a warning icon and text: 'If Loop Count is not -1 or Forever, duration will be min(Duration, Loop Count * iteration duration)', and fields for 'Duration (seconds)' and 'Startup delay (seconds)'.

2.1.3 Peticiones HTTP(S)

Las peticiones HTTP(S) son simuladas en JMeter a través de un elemento específico. Cada petición se maneja de manera independiente y es configurada para invocar a una URL específica.

La vista básica de este elemento es la siguiente:

The screenshot shows the 'HTTP Request' configuration window in JMeter. It includes fields for 'Name' (HTTP Request), 'Comments', and tabs for 'Basic' and 'Advanced'. The 'Basic' tab is active, showing 'Web Server' settings for 'Protocol [http]', 'Server Name or IP', and 'Port Number'. The 'HTTP Request' section includes 'Method: GET', 'Path', and 'Content encoding'. There are checkboxes for 'Redirect Automatically', 'Follow Redirects', 'Use KeepAlive', 'Use multipart/form-data', and 'Browser-compatible headers'. Below are tabs for 'Parameters', 'Body Data', and 'Files Upload'. The 'Parameters' tab is active, showing a table to 'Send Parameters With the Request' with columns for 'Name', 'Value', 'URL Encode?', 'Content-Type', and 'Include Equals?'.

A continuación se presentan dos ejemplos de distintas peticiones HTTP(S).

El primer ejemplo es una petición GET a una página de inicio que se encuentra ubicada en la URL <http://www.blazedemo.com>.

The image shows the 'HTTP Request' configuration dialog in Apache JMeter. The 'Name' field is set to 'HTTP Request'. The 'Basic' tab is selected, showing the 'Web Server' section with 'Protocol [http:]' set to 'http' and 'Server Name or IP' set to 'blazedemo.com'. The 'HTTP Request' section has 'Method' set to 'GET' and 'Path' is empty. There are checkboxes for 'Redirect Automatically', 'Follow Redirects', 'Use KeepAlive', 'Use multipart/form-data', and 'Browser-compatible headers'. The 'Parameters' tab is also visible, showing a table for 'Send Parameters With the Request' with columns for Name, Value, URL Encode?, Content-Type, and Include Equals?.

Esta petición simula la carga de la siguiente página:

The image shows a screenshot of the 'Simple Travel Agency' website. The header is black with the text 'Travel The World home'. The main content area has a white background with the heading 'Welcome to the Simple Travel Agency!'. Below the heading, there is a message: 'The is a sample site you can test with BlazeMeter!' and a link: 'Check out our destination of the week! The Beach!'. The form has two dropdown menus: 'Choose your departure city:' with 'Paris' selected, and 'Choose your destination city:' with 'Buenos Aires' selected. A blue button labeled 'Find Flights' is at the bottom.

Más adelante se presentará una forma de verificar en JMeter si la petición cargó correctamente. En este caso se puede observar que esto puede verificarse visualmente si aparece el texto “Welcome to the Simple Travel Agency” en la pantalla.

El segundo ejemplo es una petición POST que se genera al seleccionar el aeropuerto de salida y aeropuerto de llegada, luego de presionar el botón “Find Flights”. En este caso, el objeto que se invoca tiene el nombre “reserve.php”.

Send Parameters With the Request:					
Name:	Value	URL Encode?	Content-Type	Include Equals?	
fromPort	Paris	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>	
toPort	Buenos+Aires	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>	

Esta petición debe tener como resultado una página como se visualiza a continuación:

Choose	Flight #	Airline	Departs: Paris	Arrives: Buenos Aires	Price
Choose This Flight	43	Virgin America	1:43 AM	9:45 PM	\$472.56
Choose This Flight	234	United Airlines	7:43 AM	10:45 PM	\$432.98
Choose This Flight	9696	Aer Lingus	5:27 AM	8:22 PM	\$200.98
Choose This Flight	12	Virgin America	11:23 AM	1:45 PM	\$765.32
Choose This Flight	4346	Lufthansa	1:45 AM	8:34 PM	\$233.98

2.1.4 Manejo de cookies

Las cookies son pequeños archivos que se guardan en el cliente y contienen información relacionada al usuario en el contexto de la aplicación web que está utilizando. En las cookies es común que se almacene información sobre la sesión del usuario que está ejecutando la aplicación, así como otros datos que permiten que el usuario tenga una experiencia personalizada cuando utiliza la aplicación.

En JMeter las cookies son manejadas a través de un elemento específico que almacena y envía las cookies de igual forma que un explorador web.

Al realizar una petición HTTP(S), en la respuesta puede venir una nueva cookie, que se almacena automáticamente en este elemento en JMeter, y es enviada en cada una de las siguientes peticiones del script de prueba.

2.1.5 Manejo de caché

La caché es un importante elemento que forma parte de todos los exploradores web actuales, y permite almacenar de manera local algunos recursos, sin la necesidad de ir a buscarlos al servidor constantemente. Estos recursos son típicamente estáticos, como por ejemplo imágenes, CSS, JavaScript, entre otros.

Un buen manejo de la caché tiene un impacto positivo en el rendimiento de las aplicaciones web. Las razones principales para el almacenamiento en caché son, en primer lugar reducir los tiempos de latencia y descarga de los recursos, y por otro lado reducir el tráfico de red entre el cliente y el servidor.

JMeter permite gestionar la caché a través de un elemento específico. En este punto es importante recordar que JMeter no es un explorador web, por lo tanto se encarga de emular el comportamiento del explorador web, pero la implementación de la caché puede llegar a ser distinta en cada caso.

2.1.6 Manejo de encabezados HTTP(S)

Los encabezados HTTP(S) se envían al servidor desde el cliente, con la información adicional requerida para cumplir con requisitos específicos del servidor y de esta manera responder adecuadamente a la solicitud del cliente.

JMeter proporciona un elemento específico para adjuntar esa información adicional a la petición, o en algunos casos anular encabezados.

2.1.7 Valores por defecto de peticiones HTTP (S)

Es común que las peticiones HTTP(S) dentro de un plan de pruebas sean realizadas utilizando los mismos valores, por ejemplo en el caso del nombre del servidor y el puerto.

Para esto existe un elemento en JMeter donde se configuran estos valores y los mismos son tomados por las peticiones HTTP(S) por defecto.

Es una buena práctica utilizar este elemento, ya que en el futuro podrá ser necesario reestructurar los scripts de prueba y ejecutarlos en otro servidor. Si se cuenta con este elemento y el mismo es utilizado correctamente, realizar estos cambios es un trabajo muy sencillo

2.2 Grabación y ejecución

2.2.1 Grabación de scripts

JMeter permite grabar la navegación de un usuario interactuando con una aplicación web, lo que permite crear los scripts de prueba de manera más fácil y precisa.

Para esto se incluye en JMeter un elemento específico que se visualiza de la siguiente forma:

The screenshot shows the configuration window for the HTTP(S) Test Script Recorder. It includes fields for Name, Comments, and State. Below these are Start, Stop, and Restart buttons. The Global Settings section contains Port (8888) and HTTPS Domains. The Test Plan Creation section has tabs for Test Plan Creation and Requests Filtering. Under Test Plan Creation, there is a Target Controller dropdown set to 'Use Recording Controller', a Grouping dropdown set to 'Do not group samplers', and checkboxes for 'Capture HTTP Headers', 'Add Assertions', and 'Regex matching'. The HTTP Sampler settings section includes a Prefix dropdown, a field for 'Create new transaction after request (ms)', checkboxes for 'Retrieve All Embedded Resources', 'Redirect Automatically', and 'Use KeepAlive', and a 'Follow Redirects' checkbox. A Type dropdown is also present at the bottom.

Los pasos para realizar una grabación son los siguientes:

1. Establecer el puerto donde será realizada la grabación.
2. Seleccionar cuál será el controlador principal. Al seleccionar uno de ellos, toda la grabación quedará dentro de este elemento.
3. Hacer clic sobre la opción “Start” para iniciar la grabación.
4. El acceso a la aplicación será a través de un explorador web, por lo tanto se debe configurar para que el mismo utilice como proxy a JMeter.
5. Ahora es posible comenzar a grabar. Para esto, acceder a la aplicación web a través de la URL y comenzar a ejecutar los flujos de prueba.
6. En JMeter comenzarán a generarse las peticiones correspondientes a las interacciones con la aplicación web.
7. Dentro del grupo de hilos seleccionado, quedarán las peticiones grabadas para ser utilizadas como parte del script de prueba.

2.2.2 Recursos embebidos


Al ejecutar pruebas de rendimiento, es necesario simular el comportamiento real de los usuarios interactuando con la aplicación. Esto quiere decir, cuando se envía una petición al servidor, este devuelve la respuesta y todos los recursos asociados, que generalmente son distintos tipos de archivos, como por ejemplo imágenes, CSS, JavaScript, entre otros.

JMeter brinda la opción de excluir los recursos embebidos, para realizar grabaciones más claras y entendibles. Al excluir los recursos embebidos en las grabaciones, no quiere decir que queden excluidos en la ejecución de las pruebas. Para lograr que las pruebas sean lo más cercanas posibles al comportamiento real de la aplicación, es posible configurar las peticiones HTTP(S) para que recuperen todos los recursos embebidos.

De esta forma, el comportamiento al ejecutar el script será similar al comportamiento de un usuario utilizando un explorador web, donde se descargan todos los recursos embebidos.

2.2.3 Ejecución del script

Una vez definido el script, es importante comprender las distintas modalidades de ejecución que tiene JMeter.

El script puede ser ejecutado desde el menú “Run”, utilizando la opción “Start”, o a través de la barra de herramientas, presionando el botón .

Una vez comenzada la ejecución del script, se habilitan las siguientes opciones del menú:

- La opción “Stop” detiene la ejecución del script de manera inmediata.
- La opción “Shutdown” detiene la ejecución del script una vez que todos los elementos en ejecución terminen.

Además de habilitarse las opciones de menú para detener el script, cuando el mismo está en ejecución, es posible observar cierta información en la esquina superior derecha. Los elementos que aquí se observan son los siguientes:

Este icono  indica que el script no está siendo ejecutado.

Si el mismo se encuentra en color verde , indica que el script se encuentra en ejecución.

También es posible observar el tiempo que lleva ejecutando el script, así como los errores encontrados durante la ejecución del script, y cuantos usuarios se encuentra activos sobre la cantidad total.

2.3 Receptores de resultados

2.3.1 Introducción a los receptores

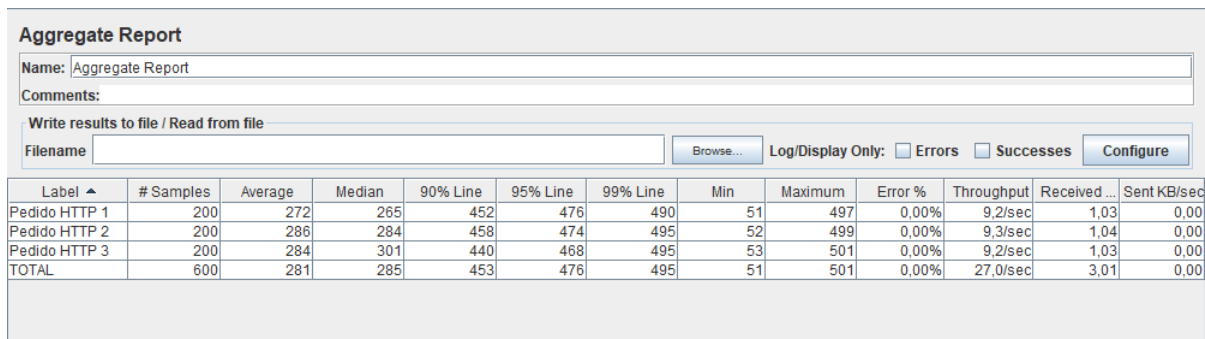
Un receptor de resultados es un componente de JMeter que muestra los resultados de las pruebas. Los resultados se pueden mostrar en un árbol, tablas, gráficos o simplemente escribirse en un archivo de registro.

La mayoría de los receptores permiten, por un lado, almacenar los resultados de las pruebas, pero también tienen la capacidad de abrir un archivo guardado previamente y visualizar los resultados en el formato del receptor.

2.3.2 Reporte resumen

Este reporte es uno de los más sencillos de JMeter, pero es muy utilizado porque permite observar en un alto nivel los resultados obtenidos.

Genera una tabla donde los resultados de las pruebas se agrupan por el nombre de las peticiones HTTP(S).



Label ^	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Maximum	Error %	Throughput	Received...	Sent KB/sec
Pedido HTTP 1	200	272	265	452	476	490	51	497	0,00%	9,2/sec	1,03	0,00
Pedido HTTP 2	200	286	284	458	474	495	52	499	0,00%	9,3/sec	1,04	0,00
Pedido HTTP 3	200	284	301	440	468	495	53	501	0,00%	9,2/sec	1,03	0,00
TOTAL	600	281	285	453	476	495	51	501	0,00%	27,0/sec	3,01	0,00

2.3.3 Árbol de resultados

Este reporte permite visualizar todas las peticiones y sus respuestas en una estructura de árbol. Además de la petición HTTP(S) y su respuesta, se pueden ver valores generales como el tiempo de respuesta, cantidad de bytes enviados y recibidos, entre otros datos.

View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename: Log/Display Only: Errors Successes

Search: Case sensitive Regular exp.

Text

- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 1
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 1
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 1
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 1
- ✓ **Pedido HTTP 1**
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 3
- ✓ Pedido HTTP 2
- ✓ Pedido HTTP 3

Sampler result Request Response data

Thread Name: Thread Group 1-6
Sample Start: 2019-08-04 19:43:38 GFT
Load time: 391
Connect Time: 5
Latency: 2
Size in bytes: 132
Sent bytes: 0
Headers size in bytes: 0
Body size in bytes: 132
Sample Count: 1
Error Count: 0
Data type ("text"|"bin"): text
Response code: 200
Response message: OK

SampleResult fields:
ContentType:
DataEncoding: null

Scroll automatically?

2.3.3 Tabla de resultados

Este reporte permite analizar los resultados en una tabla detallada, donde cada línea corresponde a cada una de las peticiones ejecutadas.

View Results in Table

Name: View Results in Table

Comments:

Write results to file / Read from file

Filename: Log/Display Only: Errors Successes

Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes	Sent Bytes	Latency	Connect Time(ms)
1	20:01:12.441	Thread Group 1-1	Pedido HTTP 1	122	✓	132	0	40	5
2	20:01:12.564	Thread Group 1-1	Pedido HTTP 2	163	✓	114	0	16	5
3	20:01:12.727	Thread Group 1-1	Pedido HTTP 3	258	✓	114	0	28	2
4	20:01:12.985	Thread Group 1-1	Pedido HTTP 1	248	✓	132	0	50	1
5	20:01:12.943	Thread Group 1-2	Pedido HTTP 1	306	✓	132	0	16	2
6	20:01:13.233	Thread Group 1-1	Pedido HTTP 2	263	✓	114	0	32	3
7	20:01:13.250	Thread Group 1-2	Pedido HTTP 2	367	✓	114	0	31	2
8	20:01:13.496	Thread Group 1-1	Pedido HTTP 3	162	✓	114	0	46	4
9	20:01:13.442	Thread Group 1-3	Pedido HTTP 1	319	✓	132	0	44	5
10	20:01:13.659	Thread Group 1-1	Pedido HTTP 1	213	✓	132	0	43	5
11	20:01:13.618	Thread Group 1-2	Pedido HTTP 3	292	✓	114	0	33	2
12	20:01:13.762	Thread Group 1-3	Pedido HTTP 2	198	✓	114	0	6	1
13	20:01:13.873	Thread Group 1-1	Pedido HTTP 2	197	✓	114	0	8	2
14	20:01:13.944	Thread Group 1-4	Pedido HTTP 1	127	✓	132	0	30	1
15	20:01:13.961	Thread Group 1-3	Pedido HTTP 3	170	✓	114	0	30	5
16	20:01:14.070	Thread Group 1-1	Pedido HTTP 3	85	✓	114	0	48	2
17	20:01:13.911	Thread Group 1-2	Pedido HTTP 1	261	✓	132	0	47	5
18	20:01:14.132	Thread Group 1-3	Pedido HTTP 1	95	✓	132	0	12	4
19	20:01:14.173	Thread Group 1-2	Pedido HTTP 2	139	✓	114	0	30	5
20	20:01:14.071	Thread Group 1-4	Pedido HTTP 2	262	✓	114	0	39	3
21	20:01:14.228	Thread Group 1-3	Pedido HTTP 2	157	✓	114	0	30	5
22	20:01:14.155	Thread Group 1-1	Pedido HTTP 1	254	✓	132	0	16	3
23	20:01:14.313	Thread Group 1-2	Pedido HTTP 3	234	✓	114	0	23	1
24	20:01:14.443	Thread Group 1-5	Pedido HTTP 1	151	✓	132	0	31	3
25	20:01:14.386	Thread Group 1-3	Pedido HTTP 3	230	✓	114	0	12	1
26	20:01:14.548	Thread Group 1-2	Pedido HTTP 1	120	✓	132	0	15	4
27	20:01:14.617	Thread Group 1-3	Pedido HTTP 1	77	✓	132	0	42	3

Scroll automatically? Child samples? No of Samples 600 Latest Sample 173 Average 274 Deviation 129

Capítulo 3 - Scripts avanzados

LO10	Comprender el concepto de correlación y cómo utilizar expresiones regulares. (K2)
LO11	Aplicar el uso de expresiones regulares para manejar la correlación en JMeter. (K3)
LO12	Comprender el concepto de parametrización. (K2)
LO13	Construir y configurar fuentes de datos para utilizar en el script de pruebas. (K3)
LO14	Comprender y aplicar temporizadores, aserciones y controladores en JMeter. (K3)
LO15	Depuración de un script en JMeter. (K3)

3.1 Correlación

3.1.1 ¿Qué es la correlación?

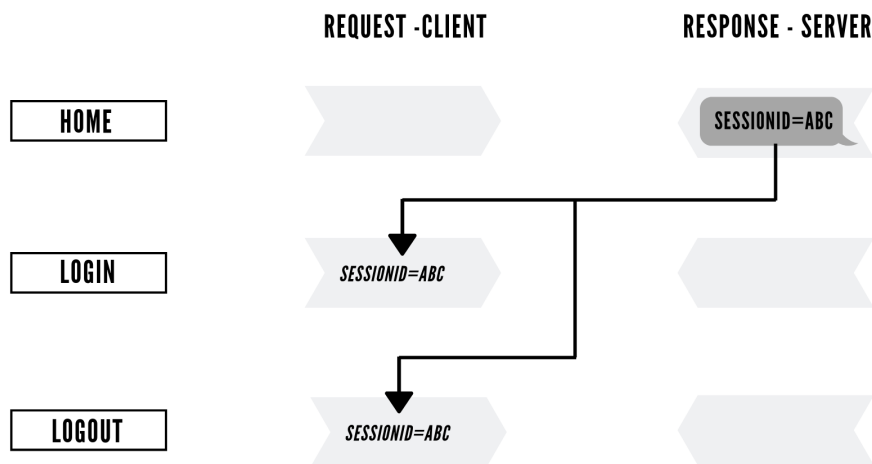
La correlación es una de las tareas más difíciles en la automatización de los scripts. Es el proceso de captura y almacenamiento de los valores dinámicos enviados desde el servidor y su inclusión en las peticiones desde el cliente.

Un valor se considera dinámico cuando puede devolver datos diferentes para cada petición en cada iteración.

Es un proceso crítico durante la ejecución de los scripts de prueba. Si no se gestiona correctamente, el script no será correctamente ejecutado.

Para llevar a cabo este proceso, primeramente, se debe identificar que valores son dinámicos, luego encontrar la respuesta en la que el servidor envía estos valores, seguidamente extraer estos valores a través de expresiones regulares, y finalmente sustituir estos valores en las peticiones desde donde son enviados de vuelta al servidor. En muchos de los casos existen varios valores dinámicos, con diferentes estructuras asociadas.

En la siguiente imagen se representa una ejecución de una prueba que consiste en ir a la página principal de la aplicación, iniciar sesión y luego cerrar la sesión. Cada vez que se realiza una ejecución, el valor de la variable “SessionID” cambiará. En el ejemplo, la variable es enviada desde el servidor en la primera petición desde la página principal, luego la aplicación estará usando este valor para ejecutar los restantes (inicio de sesión y cierre de sesión), para este caso la variable tendrá el valor “ABC”. Luego si se realiza otra ejecución de las mismas funcionalidades, la variable “SessionID” tomará otro valor. A esta situación es la que debe darle una solución el proceso de correlación.



[Traducción: En la imagen se muestra la interacción entre el usuario (client) y el servidor (server), para tres peticiones distintas (home, login, logout)]

Cuando se realiza una grabación con JMeter, se capturan los valores de cada una de las peticiones. JMeter no cuenta con un proceso de correlación automatizado, este debe ser realizado de forma manual.

En caso de no realizarlo, se enviarán los valores dinámicos como si fueran estáticos, y las pruebas no serán ejecutadas correctamente.

3.1.2 Introducción a las expresiones regulares

Uno de los elementos más importantes a tener en cuenta para el proceso de correlación son las expresiones regulares, también conocidas como “Regex”.

Una expresión regular es una secuencia de caracteres, utilizadas para la búsqueda de patrones de cadenas de caracteres u operaciones de sustituciones; estas son ampliamente usadas para extraer valores dinámicos en las respuestas enviadas

desde el servidor.

Ciertos caracteres (metacaracteres) son utilizados para indicar funciones especiales dentro del patrón, como alternativas o repeticiones, por lo que dejan de ser denominados literales, esto quiere decir que ya no representan su valor natural. Estos caracteres son: \ . ^ \$ [] () | ? + *.

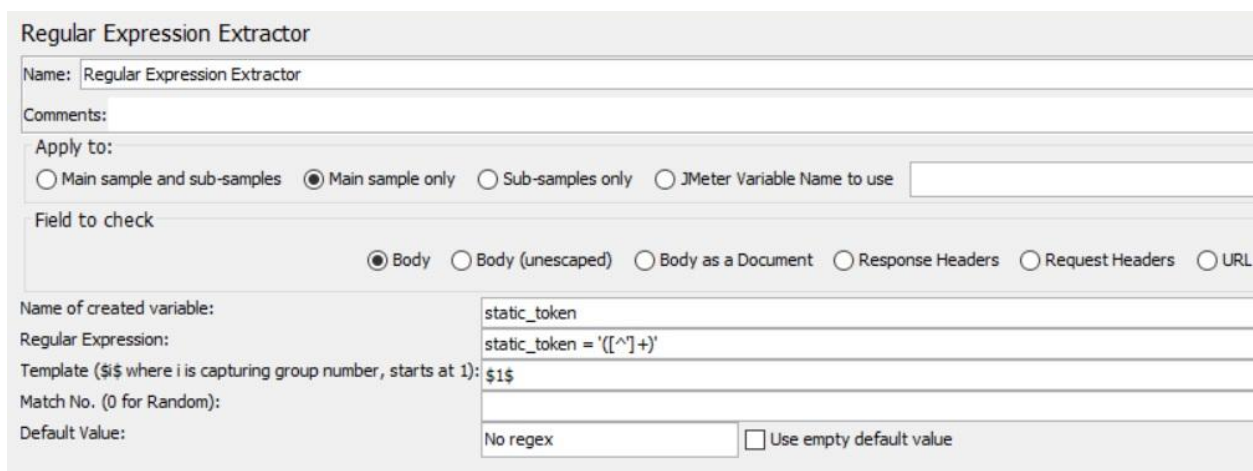
Una de las expresiones regulares más frecuentemente usada en JMeter es (.+?) debido a que existen muchos valores dinámicos con una estructura similar que pueden ser capturados con esta expresión regular. De todas formas, no siempre los valores dinámicos presentan una misma estructura.

3.1.3 Extractor de expresiones regulares

Para usar las expresiones regulares en JMeter, existe un componente extractor de expresiones regulares. Este elemento permite extraer valores de una respuesta del servidor utilizando expresiones regulares.

Este elemento será ejecutado después de cada solicitud, aplicando la expresión regular, extrayendo los valores solicitados y se almacenen los mismos en una variable específica.

En JMeter este elemento se visualiza de la siguiente forma:



The screenshot shows the configuration for a Regular Expression Extractor in JMeter. The fields are as follows:

- Name: Regular Expression Extractor
- Comments: (empty)
- Apply to: Main sample and sub-samples Main sample only Sub-samples only JMeter Variable Name to use
- Field to check: Body Body (unescaped) Body as a Document Response Headers Request Headers URL
- Name of created variable: static_token
- Regular Expression: static_token = '([^\+]*)'
- Template (\$i\$ where i is capturing group number, starts at 1): \$1\$
- Match No. (0 for Random): (empty)
- Default Value: No regex Use empty default value

3.2 Parametrización

3.2.1 Variables

Al igual que en diferentes lenguajes de programación o de scripting, en JMeter

se pueden definir variables. Estas son locales para cada hilo de ejecución. A diferencia de los lenguajes de programación, las variables en JMeter son todas del mismo tipo, variables del tipo texto.

Hay diferentes formas de definir variables en JMeter, el más común es el elemento “User Defined Variables”.

Estas variables pueden ser utilizadas en el script. Para esto, debe ponerse entre llaves “{ }” precedido de un símbolo de pesos.

Es decir, para utilizar una variable llamada “nombre”, se debe hacer de la siguiente forma: “\${nombre}”.

En tiempo de ejecución, esa variable tomará un valor que será asignado de manera dinámica.

3.2.2 Funciones

En JMeter existen distintas funciones predefinidas, que se invocan a través de nombres específicos.

Las funciones se invocan de la siguiente forma:

```
${_nombreDeFuncion(parametro1, parametro2, parametro3)}
```

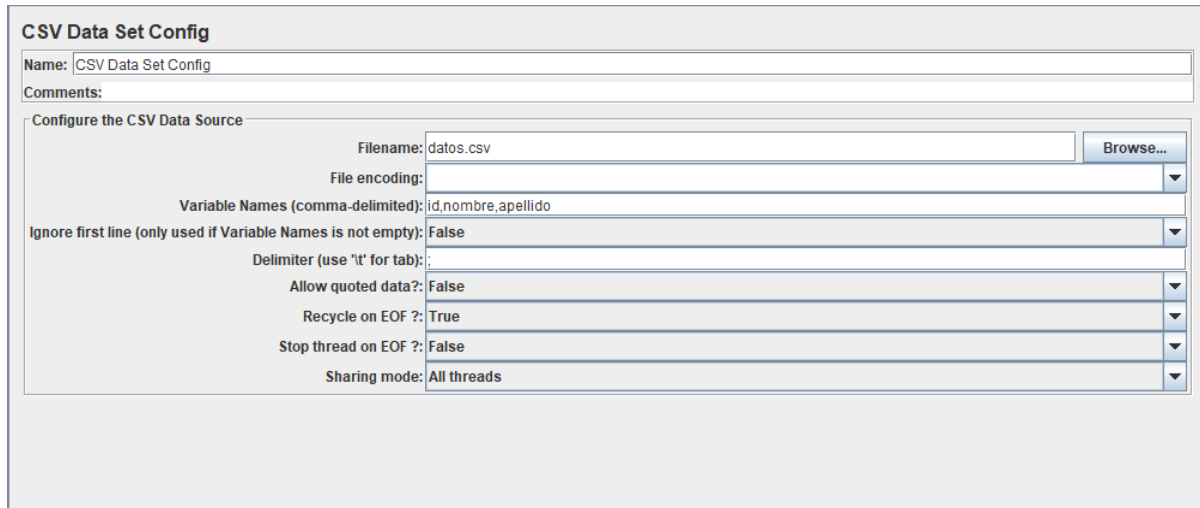
Algunas funciones necesitan de parámetros para poder funcionar. En otros casos, los parámetros son opcionales.

Las funciones se encuentran agrupadas en JMeter de acuerdo a su tipo: información, entrada, cálculo, formato, “scripting”, propiedades, variables y texto.

3.2.3 Origen de datos CSV

Este elemento permite leer datos de un archivo, que serán utilizados para las peticiones HTTP(S). Los datos se obtienen del archivo, línea por línea, y son almacenados en variables para ser utilizados en JMeter.

Este elemento se visualiza de la siguiente forma en JMeter:



CSV Data Set Config

Name: CSV Data Set Config

Comments:

Configure the CSV Data Source

Filename: datos.csv

File encoding:

Variable Names (comma-delimited): id,nombre,apellido

Ignore first line (only used if Variable Names is not empty): False

Delimiter (use '\t' for tab):

Allow quoted data?: False

Recycle on EOF?: True

Stop thread on EOF?: False

Sharing mode: All threads

El archivo “datos.csv” referenciado en el elemento anterior tiene el siguiente formato:

```
1 1;Homer;Simpson
2 2;Marge;Simpson
3 3;Lisa;Simpson
4 4;Bart;Simpson
5 5;Montgomery;Burns
6 6;Apu;Nahasapeemapetilon
7 7;Milhouse;Van Houten
8 8;Ned;Flanders
9 9;Moe;Szyslak
10 10;Nelson;Muntz
```

Por lo tanto, es posible utilizar en JMeter las variables “id”, “nombre”, “apellido” en las peticiones del script, y los datos que tomarán estas variables serán los del archivo indicado.

3.3 Tiempos de espera

3.3.1 ¿Qué son los tiempos de espera?

Cuando los usuarios utilizan una aplicación, siempre realizan pequeñas pausas entre las acciones que se ejecutan.

En las pruebas de rendimiento, el tiempo entre las distintas interacciones de usuario se denominan tiempos de espera (think time).

Los tiempos de espera juegan un papel fundamental en el script, ya que tienen como objetivo la simulación del comportamiento real de los usuarios cuando interactúan con la aplicación.

En JMeter existen elementos que se denominan temporizadores, que permiten simular los tiempos de espera.

Los temporizadores se procesan antes de cada petición, de acuerdo al nivel donde se encuentran definidos. Si hay más de un temporizador en un mismo nivel, todos los temporizadores se procesarán antes de cada petición. Un temporizador que no está en el mismo nivel que una petición no será procesado.

3.3.2 Temporizador constante

El temporizador constante es usado para agregar un tiempo de espera (retardo) constante entre cada petición. En JMeter se especifica dicho valor en milisegundos.

3.3.3 Otros temporizadores

Existen otros temporizadores que simulan distintos comportamientos.

El temporizador aleatorio uniforme retarda cada petición durante un período de tiempo aleatorio, con un máximo establecido, teniendo cada intervalo de tiempo la misma probabilidad de ocurrir.

Otro temporizador comúnmente utilizado es el gaussiano aleatorio, que permite distribuir los tiempos de respuesta aleatorios en una campaña de Gauss.

3.4 Controladores lógicos

3.4.1 ¿Qué son los controladores lógicos?

Los controladores lógicos permiten la definición de distintos flujos sobre las peticiones y el orden de las mismas.

Los conceptos manejados por JMeter son similares a los utilizados en cualquier lenguaje de programación, como por ejemplo el condicional “If”, controlador de bucles “Loop”, entre otros.

3.4.2 Controlador “If”

El controlador “If” permite controlar si los elementos de prueba debajo de él (sus hijos) se ejecutan o no.

De manera predeterminada, la condición se evalúa sólo una vez en la entrada inicial, pero existe la opción de ser evaluada para cada elemento.

Por ejemplo, si existen dos tipos de usuarios en una aplicación, un usuario básico

y un usuario administrador, donde la autenticación de cada uno es distinta, el script deberá poder manejar ambos tipos de usuarios y ejecutar las peticiones correspondientes en cada caso.

3.4.3 Controlador “Loop”

El controlador “Loop” permite ejecutar las peticiones durante un número definido de veces, o de forma indeterminada.

Por ejemplo, si se desea ejecutar 5 veces una petición determinada.

3.4.4 Controlador “Simple”

El controlador “Simple” se utiliza para organizar y almacenar las peticiones, y otros controladores lógicos. No ofrece ninguna otra funcionalidad, simplemente organizar el script para que el mismo sea más sencillo de entender y mantener.

3.4.5 Controlador “Transaction”

El controlador “Transaction” es utilizado para poder agrupar un conjunto de peticiones y otros controladores lógicos, con el objetivo de poder medir el tiempo de respuesta de la transacción que comprende el controlador.

El controlador genera una muestra adicional en los reportes, donde se indica el tiempo total de la transacción.

3.5 Aserciones

3.5.1 ¿Qué son las aserciones?

Las aserciones en JMeter son usadas para validar respuestas de peticiones que hayan sido enviados al servidor.

Las aserciones se deben ubicar como un elemento secundario dentro del script, ya sea en el plan de prueba, grupo de hilos, controladores, peticiones, entre otros.

Ayudan a verificar y asegurar que las pruebas de rendimiento están siendo ejecutadas satisfactoriamente y que el servidor bajo pruebas retorna correctamente los resultados esperados. Estas pueden ser negativas o positivas.

Por ejemplo, comprobar que la respuesta a una determinada petición contiene una cadena de texto (positiva), o comprobar que la respuesta a una determinada petición no contiene la cadena de texto (negativa).

3.5.2 Aserción de respuesta

La aserción de respuesta permite comparar cadenas de texto con la respuesta que provienen del servidor.

En JMeter este elemento se visualiza de la siguiente forma:

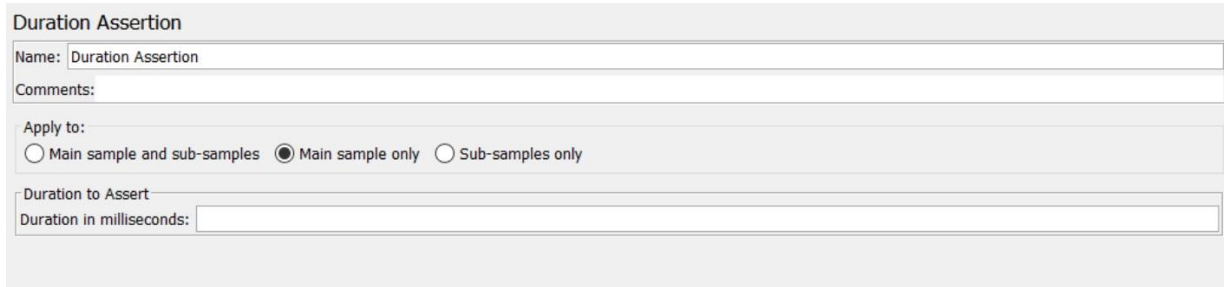
The image shows the 'Response Assertion' configuration window in JMeter. It includes fields for 'Name' (set to 'Response Assertion') and 'Comments'. The 'Apply to:' section has radio buttons for 'Main sample and sub-samples', 'Main sample only' (selected), 'Sub-samples only', and 'JMeter Variable Name to use'. The 'Field to Test' section has radio buttons for 'Text Response' (selected), 'Response Code', 'Response Message', 'Response Headers', 'Request Headers', 'URL Sampled', 'Document (text)', 'Request Data', and 'Ignore Status'. The 'Pattern Matching Rules' section has radio buttons for 'Contains', 'Matches', 'Equals', 'Substring' (selected), 'Not', and 'Or'. Below this is a 'Patterns to Test' list with 'Add', 'Add from Clipboard', and 'Delete' buttons. At the bottom, there is a 'Custom failure message' text area with a yellow highlight.

Por ejemplo, al iniciar sesión en la aplicación se muestra el texto “Bienvenido”, por lo tanto es posible definir una aserción en JMeter que verifique este comportamiento. En el caso donde la respuesta no contiene este texto, la aserción falla.

3.5.3 Aserción de duración

La aserción de duración permite verificar que las respuestas del servidor se reciban dentro de un período de tiempo determinado. Cualquier respuesta que tome más tiempo que la cantidad configurada de milisegundos se marca como una respuesta fallida.

En JMeter este elemento se visualiza de la siguiente forma:



The image shows the configuration window for a 'Duration Assertion' in JMeter. The window has a title bar 'Duration Assertion'. Below the title bar, there is a 'Name' field containing 'Duration Assertion' and a 'Comments' field. Underneath, there is an 'Apply to:' section with three radio buttons: 'Main sample and sub-samples', 'Main sample only' (which is selected), and 'Sub-samples only'. At the bottom, there is a 'Duration to Assert' section with a 'Duration in milliseconds:' label and an empty text input field.

3.6 Depuración del script

Al construir un script es posible encontrar comportamientos donde no funciona correctamente, y puede ser que la causa del error no sea sencilla de detectar. En estos casos, para identificar el error, es necesario verificar que todas las variables efectivamente tienen el valor esperado en el momento correcto.

Para esto es posible agregar un elemento específico en JMeter que permite visualizar lo que está sucediendo al ejecutar el script y de esta forma depurar el mismo.

Capítulo 4 - Ejecución de pruebas

LO16	Preparación de los scripts para la ejecución de las pruebas. (K3)
LO17	Ejecución de las pruebas utilizando el modo línea de comandos. (K3)
LO18	Ejecución de las pruebas a través del modo distribuido. (K1)
LO19	Comprender cómo se realiza el monitoreo de los recursos del sistema durante las pruebas de rendimiento y sus principales indicadores. (K2)
LO20	Aplicar herramientas básicas de monitoreo durante la ejecución de las pruebas de rendimiento. (K3)

4.1 Ejecución de las pruebas

4.1.1 Preparación del script

Existen varias actividades a realizar previo a la ejecución de las pruebas, algunas relacionadas al script en sí mismo y otras que tienen que ver con la disponibilidad del sistema y las configuraciones del monitoreo.

En relación al script, se debe tener en cuenta la limpieza de elementos utilizados para su depuración, receptores de resultados utilizados temporalmente, configuración de las variables, habilitar las peticiones secundarias, incluir tiempos de espera, entre otros aspectos.

Por otro lado, también se debe verificar que el script funciona satisfactoriamente con distintos juegos de datos, ejecutando con más de un usuario. Esto quiere decir que se deben realizar pruebas del script, antes de comenzar a ejecutar las pruebas. En este punto se debe verificar que el script ejecuta por completo y el volumen de datos a utilizar es acorde a la cantidad de hilos/usuarios e iteraciones que se desean ejecutar.

Asimismo, existen algunas configuraciones de ambiente y verificaciones sobre los datos que se recomiendan realizar previo a la ejecución de las pruebas. Por ejemplo, el ambiente debe estar disponible con todos sus componentes, de

acuerdo a la configuración establecida en el plan de pruebas. Si existen componentes que no son parte de las pruebas, es importante verificar que los mismos se encuentran aislados, aplicando el mecanismo acordado.

Como parte de las verificaciones finales, es importante realizar una prueba del monitoreo, para asegurar que las herramientas se encuentran funcionando y están configuradas para tomar datos de los indicadores definidos en el plan de pruebas.

4.1.2 Ejecución sin interfaz gráfica

Una vez que el script se encuentra finalizado, resulta más conveniente ejecutar JMeter en modo línea de comandos, ya que consume menos recursos en la máquina generadora desde donde se lanzan las pruebas.

Esto es recomendado especialmente cuando los scripts son complejos y manejan grandes volúmenes de datos.

A través de la ejecución en este modo, la integración con otras herramientas, como por ejemplo Jenkins, es muy fácil de implementar.

4.1.3 Ejecución distribuida

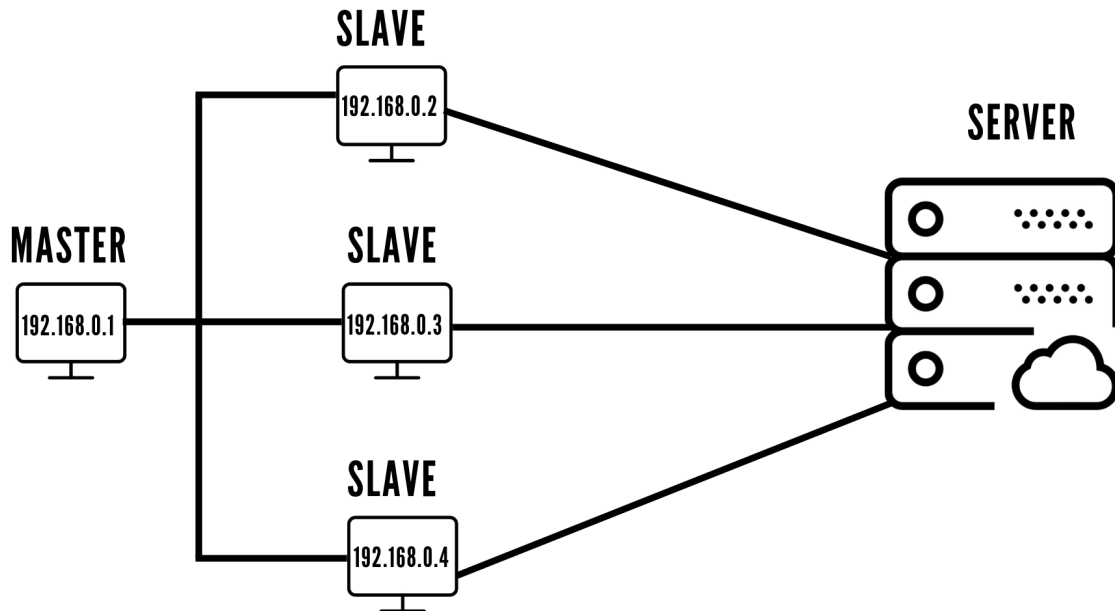
Cuando la carga de trabajo a simular es muy grande, es conveniente utilizar el modo de ejecución distribuida de JMeter.

Dado que desde una única máquina generadora de carga no es posible generar grandes cantidades de hilos/usuarios, múltiples máquinas generadoras de carga pueden ser utilizadas para esto.

Antes de profundizar en la ejecución de pruebas distribuidas, se presentan algunos términos:

- **Máster (Master):** La instancia o máquina desde donde se estará controlando las pruebas distribuidas, que se ejecutan desde otras máquinas generadoras de carga.
- **Esclavas (Slave):** Instancias o máquinas generadoras de carga.

Desde una máquina que tendrá el rol Máster, serán controladas las demás instancias Esclavas, para lograr la carga esperada sobre la aplicación.



[Traducción: En la imagen se muestra una instancia máster (master) que controla a tres instancias esclavas (slave), que son las que ejecutan las pruebas contra el servidor (server)]

4.2 Monitoreo de la aplicación

4.2.1 Introducción al monitoreo

El monitoreo es una actividad fundamental a la hora de ejecutar pruebas de rendimiento, para poder obtener información relevante que será utilizada para analizar mejoras en el rendimiento de la aplicación.

JMeter brinda información a través de los receptores de resultados. Cada uno de ellos muestra la información a través de distintos reportes, en algunos casos consolidando la información obtenida y en otros casos de forma desagregada. Es importante aclarar que esta información es obtenida en JMeter del lado del cliente, ya que es donde la herramienta se ubica para realizar la simulación de carga.

Si bien esta información es muy importante, también se requiere realizar un monitoreo del lado del servidor. En esta sección se detalla la forma de hacer esto.

Al simular carga en una aplicación web, es importante entender lo que está sucediendo a nivel de servidor y su infraestructura. Para esto es necesario definir algunos indicadores que permitan comprender si los recursos se encuentran

funcionando correctamente o si existe algún problema.

Para comprender este concepto, a modo de ejemplo, si se ejecutan pruebas de estrés sobre la aplicación y no se recibe respuesta de las peticiones HTTP(S), es importante comprender la causa de este problema.

Al realizar el monitoreo, es común detectar situaciones en donde uno de los recursos se encuentra saturado, y por lo tanto el mismo es probable que sea el “cuello de botella” en el rendimiento de la aplicación.

El monitoreo es una tarea compleja donde se recomienda un trabajo en conjunto con el equipo de infraestructura y operaciones, de manera a definir los indicadores que serán monitoreados, así como la recolección de la información durante la ejecución de las pruebas y su posterior análisis.

Es importante comprender que, idealmente, el monitoreo debe realizarse a distintos niveles de la infraestructura y del software.

4.2.2 Indicadores primarios

Los indicadores son utilizados para evaluar el estado de los distintos recursos que se están monitoreando. Los indicadores primarios permiten realizar el monitoreo de primer nivel.

Los mismos generalmente se definen para distintos componentes de la solución, entre ellos:

- Servidores (a nivel de sistema operativo)
- Servidores de aplicación (como por ejemplo servidores web)
- Servidores de base de datos
- Redes
- Aplicación (indicadores específicos de la aplicación)

Los recursos que comúnmente son monitoreados a través de indicadores primarios para los servidores son:

CPU: Como indicadores primarios para el recurso CPU, se puede observar el porcentaje de utilización de la misma, así como el largo de la cola de trabajos por procesar.

Memoria: Para el monitoreo de la memoria, es importante conocer la capacidad del servidor y observar la memoria disponible durante las pruebas.

Disco: Para el monitoreo del disco, es importante observar la actividad de lectura y escritura en el mismo. Asimismo, se recomienda observar el largo de la cola de trabajos para el disco y el espacio disponible del mismo.

Red: A nivel de interfaz de red, es relevante observar la actividad de envío y recepción de datos. Se recomienda conocer la capacidad de los enlaces utilizados para las pruebas, entre los distintos componentes de la solución.

4.2.3 Herramientas básicas de monitoreo

Si bien existen una variedad muy grande de herramientas para llevar a cabo el monitoreo durante las pruebas, existen dos herramientas que son muy útiles para los dos sistemas operativos más utilizados.

4.2.3.1 Windows Performance Monitor

Para soluciones en las que el servidor de aplicaciones ejecuta sobre un sistema operativo Windows, es posible utilizar la herramienta “Windows Performance Monitor”, también conocida como “perfmon”.

Con esta herramienta es posible configurar diferentes indicadores, los cuales están agrupados por categoría.

4.2.3.2 NMon

Por otro lado, para las soluciones cuyo servidor de aplicaciones ejecuta sobre sistemas operativos Linux, es posible utilizar la herramienta “NMon”.

Dicha herramienta es del tipo de línea de comando, y tiene dos modos de ejecución.

El primer modo se utiliza para observar la actividad de los indicadores en tiempo real, y se activa solamente ejecutando el comando “nmon” en la terminal. Al iniciar muestra distintas opciones de indicadores para visualizar.

El segundo modo es para almacenar la actividad de los indicadores en un archivo que luego podrá ser analizado.

Capítulo 5 - Documentación

LO21	Documentación sobre las pruebas de rendimiento. (K2)
------	--

5.1 Introducción

En el proceso de pruebas de rendimiento, es importante la definición de algunos documentos, que brinden apoyo a la ejecución de las actividades y permiten el registro de los resultados obtenidos.

Los principales documentos son:

- Plan de pruebas de rendimiento
- Guión de pruebas
- Informe de resultados

5.2 Plan de pruebas de rendimiento

El plan de pruebas de rendimiento es el documento que guía el proceso de pruebas, donde se plasman las principales decisiones y acuerdos.

El mismo se encuentra compuesto por las siguientes secciones:

- Introducción
- Marco de trabajo
 - Alcance
 - Cronograma de actividades
 - Organización del equipo
 - Hitos y entregables
- Plan de pruebas
 - Sistema bajo pruebas
 - Arquitectura del sistema
 - Interfaces con otros sistemas
 - Análisis de riesgos/criticidad
 - Ambiente e infraestructura de pruebas
 - Integración con sistemas externos
 - Criterios de aceptación
 - Escenarios de pruebas
 - Datos de pruebas
 - Herramientas de generación de carga
 - Monitorización

- Herramientas de monitorización
- Indicadores de monitorización

5.3 Guión de pruebas

El guión de pruebas contiene la información detallada de cada una de las pruebas con sus pasos, que se ejecutará en el sistema.

Cada guión de pruebas contiene la siguiente información:

- Objetivo
- Usuario que lo ejecuta
- Pasos para la ejecución (se presenta en formato tabla)
 - Detalle de cada paso
 - Validaciones de cada paso
 - Tiempo de espera
- Resultado esperado a nivel global

5.4 Informe de resultados

El informe de resultado de las pruebas contiene la siguiente información:

- Resumen de las pruebas realizadas
- Resultados de las pruebas
 - Bitácora de ejecuciones
 - Línea base
 - Detalle de los escenarios ejecutados
 - Información de monitoreo de cada escenario
- Mejoras aplicadas durante las pruebas
- Limitantes
- Conclusiones
- Recomendaciones

Capítulo 6 - Extra

LO22	Comprender las mejores prácticas a la hora de utilizar JMeter. (K2)
LO23	Ejecución de scripts para servicios web. (K3)

6.1 Mejores prácticas de JMeter

Existe un conjunto de buenas prácticas a la hora de utilizar JMeter, que se listan a continuación:

1. Utilizar la última versión de JMeter.
2. Identificar el número correcto de hilos/usuarios.
3. Aplicar los elementos adecuados de JMeter para cada función específica.
4. Comenzar grabando el script en JMeter.
5. Utilizar herramientas de tipo proxy para el proceso de correlación.
6. Centralizar y estandarizar las configuraciones repetidas.
7. Utilizar tiempos de espera para lograr escenarios realistas.
8. Verificar las respuestas con aserciones.
9. Reducir el consumo de recursos en la generadora de carga.

6.2 Pruebas de servicios web

Un servicio web es un conjunto de protocolos y estándares utilizados para intercambiar datos entre aplicaciones de software. Estos permiten a las organizaciones intercambiar datos sin necesidad de conocer los detalles de sus respectivas aplicaciones.

A modo de resumen, los servicios web pueden ser trabajados en JMeter como peticiones HTTP(S), incluyendo los encabezados y el formato del cuerpo correspondiente al servicio que se está invocando.

Glosario

Dado que muchos de los términos utilizados en las pruebas de rendimiento y en el uso de JMeter se presentan por defecto en inglés, se incluye el siguiente glosario que permite lograr una mejor comprensión del presente documento.

Inglés	Español
Performance	Rendimiento
Assertion	Aserción
Correlation	Correlación
Parameterization	Parametrización
Test plan	Plan de pruebas
Timer	Temporizador
Controller	Controlador
Script	Script
Debug	Depuración
Thread group	Grupo de hilos
Manager	Gestor
Header	Cabecera / Encabezado
Record	Grabación
Execution/Running	Ejecución
Resource	Recurso
Embedded	Embebido
Listener	Receptor
Regular expression (Regex)	Expresión regular
Extractor	Extractor

Think time	Tiempo de espera
Browser	Explorador / Navegador
Request	Petición
Response	Respuesta
IDE	Entorno de desarrollo integrado
Plug-in	Complemento
Render	Dibujado
GUI (Graphical User Interface)	Interfaz de usuario
Command line	Línea de comandos

Referencias

- Sitio web oficial de Apache JMeter: <https://jmeter.apache.org>
- Wiki oficial de Apache JMeter: <https://wiki.apache.org/jmeter>
- Referencia rápida de expresiones regulares: <https://docs.microsoft.com/en-us/dotnet/standard/base-types/regular-expression-language-quick-reference>
- Mozilla Reference Protocolo HTTP: <https://developer.mozilla.org/en-US/docs/Web/HTTP>
- Sitio oficial NMon: <http://nmon.sourceforge.net/>
- Guía de uso de Windows Performance Monitor: <https://techcommunity.microsoft.com/t5/Ask-The-Performance-Team/Windows-Performance-Monitor-Overview/ba-p/375481>