



Eusko Jaurlaritzaren Informatika Elkarte  
Sociedad Informática del Gobierno Vasco

1

## Apache JMeter:

Manual de usuario

Fecha:

Referencia:

EJIE S.A.  
Mediterráneo, 3  
Tel. 945 01 73 00\*  
Fax. 945 01 73 01  
01010 Vitoria-Gasteiz  
Posta-kutxatila / Apartado: 809  
01080 Vitoria-Gasteiz  
[www.ejje.es](http://www.ejje.es)

Este documento es propiedad de EJIE, S.A. y su contenido es confidencial. Este documento no puede ser reproducido, en su totalidad o parcialmente, ni mostrado a otros, ni utilizado para otros propósitos que los que han originado su entrega, sin el previo permiso escrito de EJIE, S.A.. En el caso de ser entregado en virtud de un contrato, su utilización estará limitada a lo expresamente autorizado en dicho contrato. EJIE, S.A. no podrá ser considerada responsable de eventuales errores u omisiones en la edición del documento.

## Control de documentación

Título de documento: JMeter

### Histórico de versiones

Código:

Versión: 1.3

Fecha:

Resumen de cambios: Cambios por nueva versión

### Cambios producidos desde la última versión

Primera versión.

### Control de difusión

Responsable:

Aprobado por:

Firma:

Fecha: 22/12/2014

Distribución:

### Referencias de archivo

Autor: Consultoría de áreas de conocimiento

Nombre archivo: Apache JMeter. Manual de usuario v1.3.doc

Localización:

## Contenido

	Capítulo/sección	Página
1	Introducción	4
2	Conceptos básicos	4
3	Funciones elementales	5
3.1	Plan de Pruebas	8
3.2	BadBoy	10
3.3	Pruebas Distribuidas	11
3.4	Uso del Monitor de Planes de Prueba	12
4	Integración con otros programas	15
5	Utilidad práctica	15
6	Anexo 1 – Ejemplo	15
6.1	Resolución	15
7	Parametrización : JMeter – Script Badboy XLNets	28
7.1	Crear el script de BadBoy	28
7.2	Configurar las cookies en JMeter	28

## 1 Introducción

El presente documento describe cuáles son las tareas básicas que se pueden ejecutar en la explotación de la herramienta de Carga Apache JMeter.

El contenido del documento integra, tanto los aspectos de uso en el entorno de EJIE como las características elementales de funcionamiento de la aplicación.

## 2 Conceptos básicos

Apache JMeter es una herramienta de carga diseñada para realizar Pruebas de Rendimiento y Pruebas Funcionales sobre Aplicaciones Web.

Desarrollado por THE APACHE SOFTWARE FOUNDATION, la primera versión (v1.0) data de marzo del 2001, y continúa su desarrollo hasta la actualidad.

La página de Referencia donde se puede encontrar toda la información sobre la aplicación es:  
<http://jakarta.apache.org/jmeter/>

Originalmente el Apache *JMeter* fue diseñado para realizar pruebas de estrés sobre aplicaciones web (pruebas web clásicas). Sin embargo hoy en día su arquitectura ha evolucionado, ahora no sólo puede llevar a cabo pruebas en componentes típicos de Internet (HTTP), sino también puede realizar pruebas sobre Bases de Datos, scripts Perl, servlets, objetos java, servidores FTP y prácticamente cualquier medio de los que se pueden encontrar en la red.

Para un óptimo desarrollo de pruebas, es necesario tener ciertas nociones funcionales de la aplicación que se va a evaluar. Si esto no es así, las pruebas no serán completas al no saber por ejemplo si ha devuelto la hoja apropiada a la petición hecha o si nos ha permitido acceder con un login no apropiado.

El Apache *JMeter* esta diseñado para desarrollar diferentes tipos de test; permitiendo diseñar tanto sencillos teses que soliciten simples páginas web, como complejas secuencias de requisiciones que permitan evaluar el comportamiento de una aplicación o como la capacidad de carga máxima que pueda tener una aplicación en un servidor (pudiendo llegar a satura el servidor).

*JMeter* también permite la ejecución de pruebas distribuidas entre distintos ordenadores, para realizar pruebas de rendimiento.

El Apache *JMeter* incluye una interfaz grafica de usuario que facilita el diseño de las pruebas. Este interfaz grafico además de aportar un entorno cómodo de trabajo, también permite guardar y alterar tanto los test desarrollados como los componentes que lo integran. Gracias a esto se pueden reutilizar las pruebas o módulos de las mismas en el desarrollo de nuevas pruebas.

Además de las funcionalidades de prueba antes mencionadas, el Apache *JMeter* también ofrece la posibilidad de activar un Proxy web. Gracias a esto se puede grabar la navegación de un usuario para posteriormente usarla en la generación de una prueba.

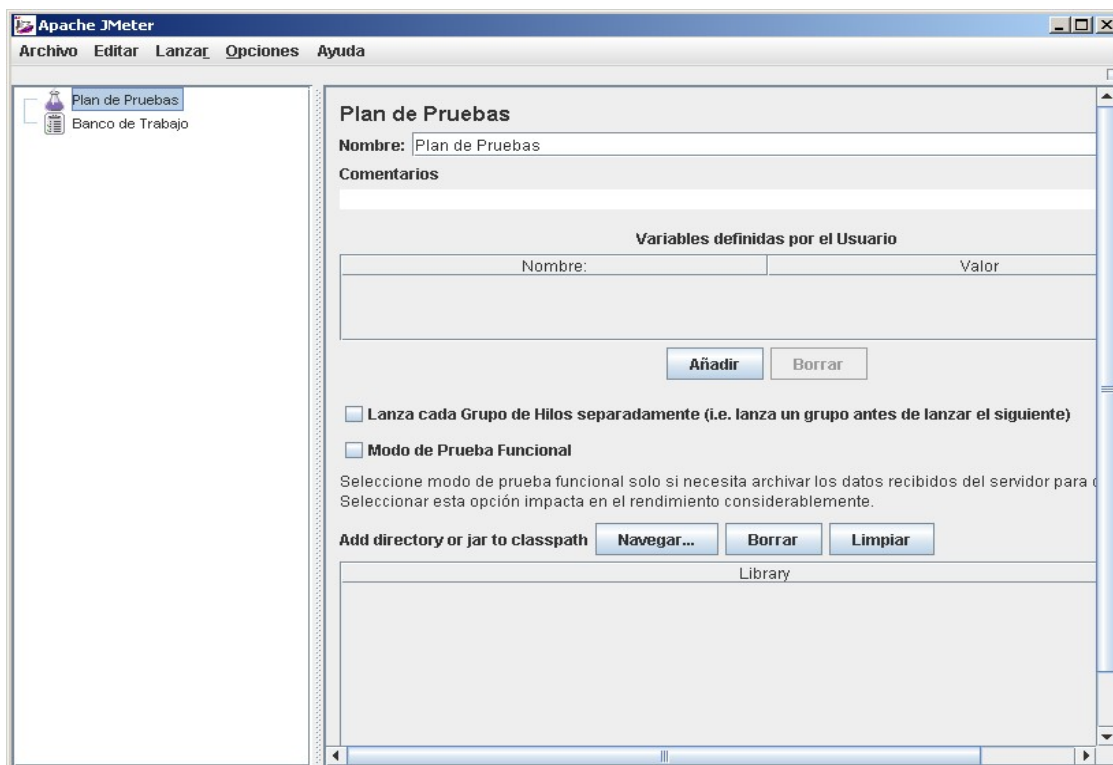
### 3 Funciones elementales

El Jakarta Jmeter permite trabajar en varios modos, pero para trabajar cómodamente la mejor opción es usar el interfaz gráfico. Para lanzarlo, ejecutaremos el `jmeter` o el `jmeter.bat` (situados en el directorio `bin` de la aplicación) según la plataforma (unix y Windows respectivamente) en la que estemos ejecutando la aplicación. De esta manera accederemos al interfaz gráfico del Apache *JMeter*.

Una vez se ejecuta el Apache *JMeter*, lo primero que hace la aplicación es buscar las clases de los `jar`'s incluidos dentro de los directorios `/lib` y `/lib/ext`. Si se quisiera añadir algún fichero `jar` (por ejemplo: el necesario para hacer una conexión a una base de datos) se debe incluir en el directorio `/lib`. Si se desearía incluir componentes específicos que sean ejecutados por *JMeter* (por ejemplo pruebas Junit), tendremos que incluir sus `jar`'s correspondientes en el directorio `/lib/ext` (para más información ver en la sección [classpath](#) de la documentación del producto).

Todas las incidencias detectadas por el Apache *JMeter* se escriben en un fichero `.log`. El nombre del fichero `log` se define en el fichero `jmeter.properties` (fichero que se encuentra en el directorio `/bin`), por defecto dicho fichero se llama `jmeter.log` y se escribe en el directorio desde el que se lanza la aplicación.

Una vez se ejecuta el fichero `jmeter` o el `jmeter.bat` según la plataforma (unix y Windows respectivamente) en la que se trabaje, la primera pantalla que se presenta es:



Esta es la pantalla principal del Apache *JMeter*. Echándole un primer vistazo, se puede ver que la pantalla tiene un formato similar al típico de las ventanas de Windows y que está dividida en dos partes fácilmente diferenciables. La parte izquierda representa la estructura o representación en árbol del plan de pruebas, mientras que en la parte derecha tendremos la plantilla de edición del elemento que tengamos elegido a la

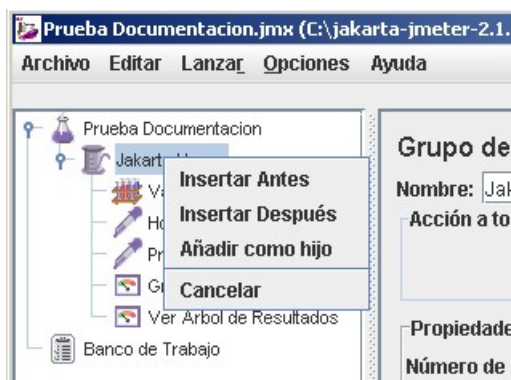
izquierda.

La aplicación tiene dos entidades básicas. Por un lado tenemos el Test plan (o plan de pruebas) que representa una prueba y por el otro los elements (o elementos) que representan las diferentes etapas o acciones que usaremos para componer las pruebas.

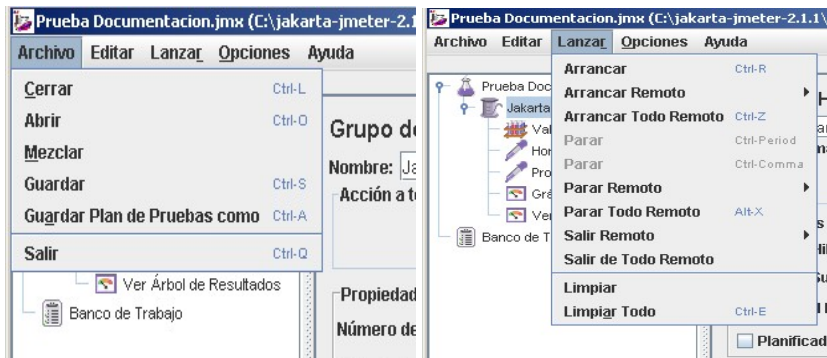
El Apache *JMeter* basa su funcionamiento en elementos integrados dentro de una estructura de árbol (dentro de un plan de pruebas). Cualquier parte (elemento) del árbol puede ser agregada o guardada de manera independiente, de tal manera que se puede reutilizar elementos en planes de pruebas posteriores. Ni que decir tiene, que también se pueden guardar los planes de pruebas enteros.

Con esta forma de trabajo el entorno gráfico es apropiado para ver y editar los elementos que componen un plan de prueba (árbol de la izquierda). Otro detalle que se debe tener muy presente, es que el orden en el árbol (orden descendente) determina el orden de ejecución de los elementos dentro de la prueba.

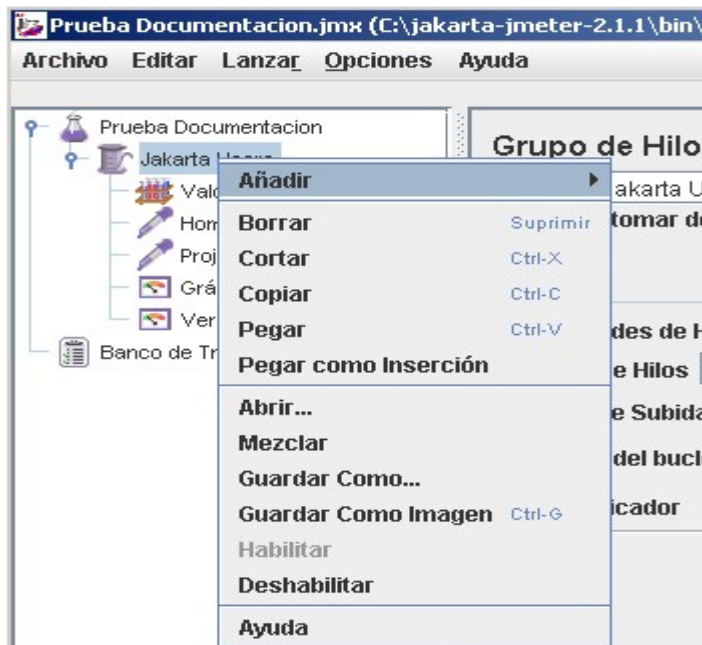
En concordancia con esto, para agilizar la configuración del orden de los elementos dentro del árbol el entorno grafico permite arrastrar (pinchando el elemento oportuno con el botón derecho y no soltando movemos el ratón arrastrando con el elemento) los distintos elementos hasta las posiciones que se consideren oportunas. Al depositar el elemento arrastrado el entorno grafico nos asistirá con un menú para especificar la ubicación exacta.



Con respecto a los menús del entorno grafico, decir que cuenta con cinco menús (archivo, edición, lanzar, opciones y ayuda). Los cuales tienen las funcionalidades básicas de manipulación de ficheros y las de funcionalidad de la aplicación. Su uso no entraña ninguna dificultad (las funcionalidades son bastante intuitivas).



Para agilizar el manejo de la aplicación, el entorno gráfico incorpora en cada elemento la opción de sacar un menú específico. Para acceder a este menú bastara con posicionarnos sobre el elemento y presionar el botón derecho del ratón.



### 3.1 Plan de Pruebas

El plan de pruebas es el objeto que representa una prueba, el plan de pruebas se compone de elementos. Estos elementos, según su funcionalidad dentro de las pruebas, los podemos dividir en ocho familias:

- [Thread Group](#) (Grupo de hilos): Elemento que define el número de hilos (threads) de ejecución que definirán el número de usuarios a simular.
- [Samplers](#) (Muestreadores): Elementos que representan las solicitudes a un servidor.
- [Logic Controllers](#) (Controladores Lógicos): Elementos lógicos que nos permiten controlar el comportamiento de las pruebas.
- [Listeners](#): Elementos que nos permiten tener acceso a los datos recopilados por las pruebas; estos pueden ser desde datos estadísticos hasta gráficas de evolución.
- [Timers](#) (Temporizadores): Elementos destinados a controlar los espacios relativos de tiempo entre los diferentes hilos (threads).
- [Assertions](#) (Aserciones): Conjunto de elementos encargados de la verificación de los datos provenientes de los servidores.
- [Configuration Elements](#) (Elementos De Configuración): tipo de elementos destinados a labores de configuración (en muchos casos valores por defecto).
- [Pre-processor Elements](#) (Pre procesadores): Los pre-procesos ejecutan una acción antes de la ejecución de un Sampler.
- [Post-Processor Elements](#) (post procesadores): Los post-procesos ejecutan una acción después de la ejecución de un Sampler.

Cada una de las familias de elementos esta compuesta varios elementos (la única excepción es el grupo de hilos que solo integra un elemento), cada uno de estos elementos realiza una función específica. Los elementos que integran las diferentes familias son los siguientes:

#### - [Samplers](#)

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

#### - [Logic Controllers](#)

- [Interleave Controller](#)
- [Loop Controller](#)
- [Once Only Controller](#)

#### - [Configuration Elements](#)

- [HTTP Authorization Manager](#)
- [HTTP Cookie Manager](#)
- [HTTP Proxy Server](#)
- [HTTP Request Defaults](#)
- [FTP Request Defaults](#)
- [JDBC Connection Configuration](#)
- [JDBC SQL Query Defaults](#)
- [Mail Reader Sampler](#)
- [HTTP Header Manager](#)
- [Login Config Element](#)
- [Simple Config Element](#)
- [LDAP Request Defaults](#)
- [LDAP Extended Request Defaults \(ALPHA\)](#)
- [Java Request Defaults](#)
- [User Defined Variables](#)
- [TCP Sampler Config](#)
- [CSV Data Set Config](#)
- [JNDI Default Configuration](#)

#### - [Listeners](#)

- [Mailer Visualizer](#)
- [Graph Full Results](#)
- [Graph Results](#)



- [Simple Controller](#)
- [Random Controller](#)
- [Recording Controller](#)
- [Module Controller](#)
- [Throughput Controller](#)
- [If Controller](#)
- [Random Order Controller](#)
- [ForEach Controller](#)
- [Transaction Controller](#)
- [Runtime Controller](#)
- [While Controller](#)
- [Switch Controller](#)
- [Include Controller](#)
- [Assertions](#)
  - [Response Assertion](#)
  - [Duration Assertion](#)
  - [Size Assertion](#)
  - [XML Assertion](#)
  - [BeanShell Assertion](#)
  - [MD5Hex Assertion](#)
  - [HTML Assertion](#)
  - [XPath Assertion](#)
  - [XML Schema Assertion](#)
- [Timers](#)
  - [Constant Timer](#)
  - [Gaussian Random Timer](#)
  - [Uniform Random Timer](#)
  - [Constant Throughput Timer](#)
  - [Synchronizing Timer](#)
- [Spline Visualizer](#)
- [Assertion Results](#)
- [View Results Tree](#)
- [Aggregate Report](#)
- [View Results in Table](#)
- [Simple Data Writer](#)
- [Monitor Results](#)
- [Distribution Graph \(alpha\)](#)
- [Aggregate Graph](#)
- [Pre Processors](#)
  - [HTML Link Parser](#)
  - [HTTP URL Re-writing Modifier](#)
  - [HTML Parameter Mask](#)
  - [HTTP User Parameter Modifier](#)
  - [User Parameters](#)
  - [Counter](#)
- [Post-Processors](#)
  - [Regular Expression Extractor](#)
  - [Result Status Action Handler](#)
  - [Save Responses to a file](#)
  - [Generate Summary Results](#)

Para obtener mayor detalle de los mismos, cada elemento incorpora un enlace al apartado de la documentación de la aplicación que los describe.

Una vez se conocen todos los elementos que se pueden usar en la creación de un plan de pruebas, ya sólo queda empezar a crearlos. Un buen punto de partida, es empezar por los [planes](#) descritos en la documentación de la aplicación ([Building a Web Test Plan](#) y [Building an Advanced Web Test Plan](#)). Estos dos ejemplos, son un buen referente para comenzar a trabajar con la aplicación.

## 3.2 BadBoy

El Badboy es una aplicación desarrollada por **Badboy software** (<http://www.badboy.com.au/>), diseñada para grabar navegaciones Web y usarlas en pruebas de estrés sencillas.

Esta aplicación incorpora una opción de exportar las grabaciones Web efectuadas por el mismo, al lenguaje de scripting interpretado por el JMeter. La aplicación es apropiada para realizar las grabaciones que posteriormente se usarán para generar las pruebas del JMeter, ya que es muy sencilla de usar y deja unas grabaciones más limpias que las realizadas por el JMeter.

Si se desea más información acerca de la instalación y uso del Badboy remitirse a los documentos:

- Badboy, Manual de Instalación en Cliente.
- Badboy, Manual Rápido de Usuario.

### 3.3 Pruebas Distribuidas

Una vez generados los planes de prueba y tras ver cómo responde el servidor frente a ellos, quizá se desee dar un paso más y ver cómo actúa el servidor frente a clientes provenientes de distintos orígenes.

La opción más sencilla es lanzar la misma prueba desde varias máquinas a la vez, pero ¿y si se desea lanzar las pruebas a la vez para ver cómo responde el servidor frente a peticiones simultáneas desde diferentes máquinas? o ¿si la prueba dura relativamente poco y para cuando se lanza el segundo plan de pruebas desde otra máquina el primero ya ha terminado? Esta solución resulta un poco engorrosa y no representa una opción demasiado apetecible si desea probar la misma prueba desde un número elevado de máquinas.

Pensando en este sentido, el Apache *JMeter* presenta un formato de ejecución remoto que permite lanzar una prueba desde distintas máquinas sincronizadamente. Esto permite realizar pruebas más completas y darles una connotación más real.

El proceso de ejecución remota lleva una serie de etapas:

#### **1ª Etapa.** Iniciar los clientes remotos:

Lo primero que se debe hacer es lanzar el Apache *JMeter* en todas las máquinas que se van a usar como clientes menos una, pero en lugar de ejecutar el fichero habitual se usará el `jmeter-server` | `jmeter-server.bat` (según sistema operativo). Con esto los clientes se quedarán esperando a recibir las instrucciones de la máquina que no se ha lanzado en ese modo.

#### **2ª Etapa.** Especificar los clientes en modo servidor al cliente principal:

El segundo paso es configurar la máquina que se usará para gestionar la prueba. A esta máquina le se le especifica las direcciones de las otras máquinas. Para hacer esto se edita el fichero de configuración `jmeter.properties` (que se encuentra en el directorio `bin/`) y se añade en el campo "remote\_hosts" los sucesivos host's de las máquinas que tenemos en espera.

#### **3ª Etapa.** Iniciar las pruebas:

Ahora ya se está preparado para ejecutar el `jmeter` | `jmeter.bat` (según sistema operativo) en la máquina que gestionará las pruebas. Al cargar el entorno gráfico observaremos que el menú `run` (lanzar) tiene dos sub-menús remotos (arrancar y parar). En estos sub-menús ahora aparecen los host's que antes se han añadido; estos host's valdrán para controlar la ejecución del plan de prueba en esas máquinas de forma remota (según se lance o se pare el plan de prueba en esas máquinas).

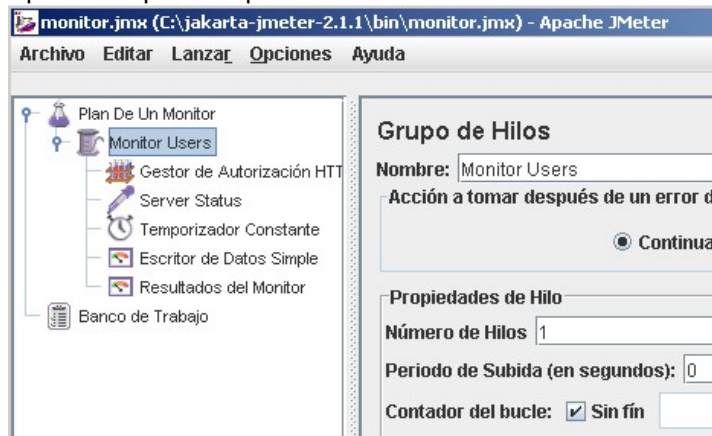
### 3.4 Uso del Monitor de Planes de Prueba

Otra de las funcionalidades que incorpora el Apache *JMeter* es el monitor de planes de pruebas. Esta funcionalidad pretende llevar un control continuo del estado del servidor en las sucesivas etapas de nuestros planes de pruebas.

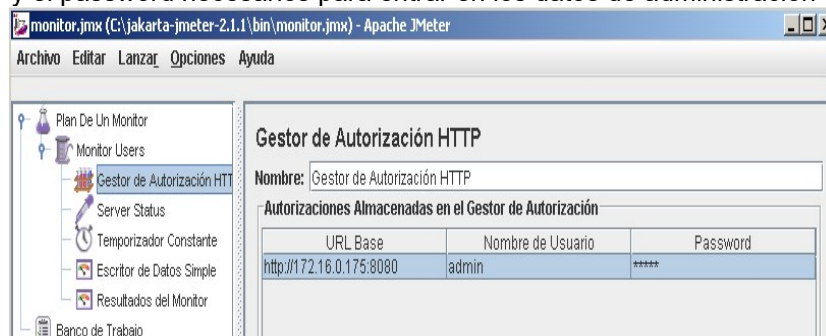
En principio esta funcionalidad fue diseñada para trabajar con el estado de los servlets de Tomcat 5. Teóricamente no deberíamos tener ningún problema con servlets de otro tipo mientras sus containers servlets soporten JMX (Java Management Extensión).

Los pasos que se deben dar para crear un monitor de planes de pruebas son los siguientes:

1. Se debe crear un nuevo test plan (plan de pruebas) y añadirle un Thread Group (Grupo de hilos) configurándolo con un solo hilo (sólo se necesita uno para ver el estado del servidor) con la opción de que se repita sin fin activada.

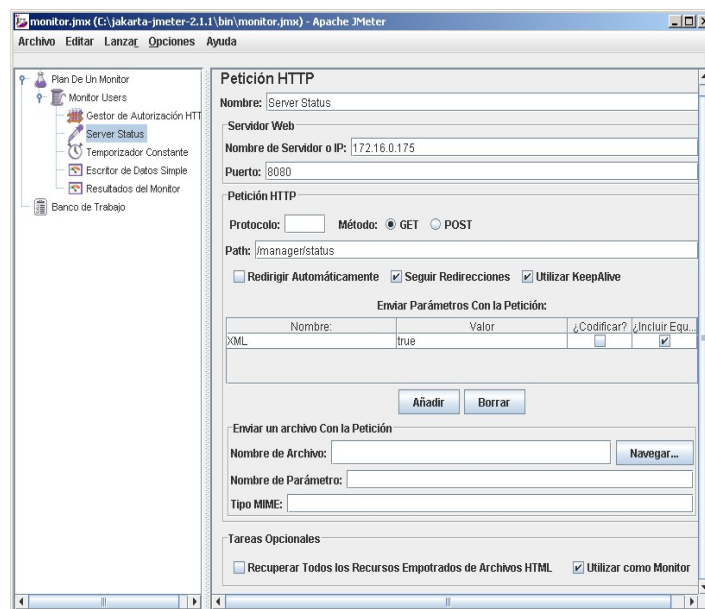


2. Una vez definido el grupo de hilos, lo siguiente es añadir dentro del controlador un HTTP Authorization Manager (Gestor de Autorización http que encontraremos en añadir→ elemento de configuración→ Gestor de Autorización http). Este objeto realiza la gestión del nombre de usuario y el password necesarios para entrar en los datos de administración del servidor.



3. Con el gestor HTTP Authorization Manager (Gestor de Autorización http) configurado se podrá acceder a los datos administrativos mediante una HTTP Request (petición HTTP que encontraremos en añadir→ muestreador→ petición HTTP). Dicho elemento lo tenemos que añadir dentro del Thread Group (Grupo de hilos) y se configurará con los siguientes datos:
  - Se introduce el nombre que consideremos oportuno (ej: "estado del servidor").
  - Se introduce la dirección IP o el Hostname del servidor.
  - Se introduce el número del puerto.
  - Se introduce el path oportuno de acceso al estatus del servidor (ej: "/manager/status" si usamos Tomcat).
  - Se añade un parámetro de la petición llamado "XML". Dicho parámetro necesita tener como valor "true".
  - Se Activa la opción de utilizar como monitor.

Configurando así la HTTP Request (petición HTTP) tendrá un aspecto parecido a éste:

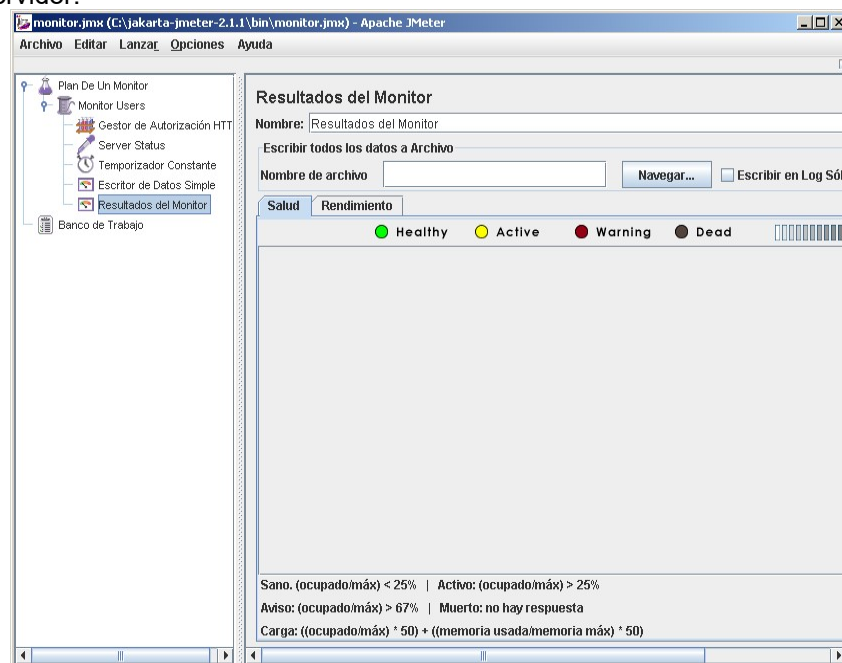


4. Para el buen funcionamiento del monitor (y para no ahogar a nuestro servidor) es necesario añadir un timer (temporizador que encontraremos en añadir→ temporizador→ temporizador constante) al Thread Group (Grupo de hilos). Con esto se consigue que las peticiones no sean continuas, sino que sean cada cierta cantidad de tiempo (un valor óptimo son 5 segundos). Con añadir un Constant Timer (temporizador constante) será suficiente.

5. Por ultimo, lo único que queda es añadir al Thread Group (Grupo de hilos) un Simple Data Writer (escritor de datos simple que se encuentra en añadir→ listener→ escritor de datos simple) para copiar los datos extraídos en un fichero



y un Monitor Results (resultados del monitor que podemos encontrar en añadir→ listener→ resultados del monitor) para visionar en tiempo de ejecución los datos correspondientes al estado del servidor.



El Monitor Results (resultados del monitor) incorpora dos tabs (salud y rendimiento) donde se puede ver el estado general del servidor con cada petición y la gráfica representativa del estado del servidor. Esta última gráfica ira actualizándose a razón del tiempo configurado en el timer (temporizador).

## 4 Integración con otros programas

Además del entorno gráfico de trabajo, el Apache *JMeter* presenta alternativas de ejecución. Estos formatos de ejecución están diseñados para posibilitar la integración del Apache *JMeter* con otras aplicaciones.

Los formatos de ejecución que presenta el Apache *JMeter*, además del entorno gráfico, son:

- Lanzar la aplicación en modo no-gráfico. La aplicación es lanzada mediante la línea de comandos (o desde otra aplicación) con unos parámetros determinados ([ver documentación](#)).
- La aplicación está adaptada para ser usada desde una tarea Ant ([ver jmeter ANT task](#)).
- El Apache *JMeter* tiene en desarrollo un plug-in para Eclipse.

## 5 Utilidad práctica

JMeter es una herramienta que sirve para realizar pruebas de rendimiento y de funcionalidad sobre aplicaciones tipo cliente/servidor.

Por tanto será útil durante la fase de desarrollo de las aplicaciones para la realización de pruebas funcionales y de regresión.

Además nos será de gran utilidad durante la fase de pruebas para la realización de pruebas de carga y de volumen de nuestras aplicaciones.

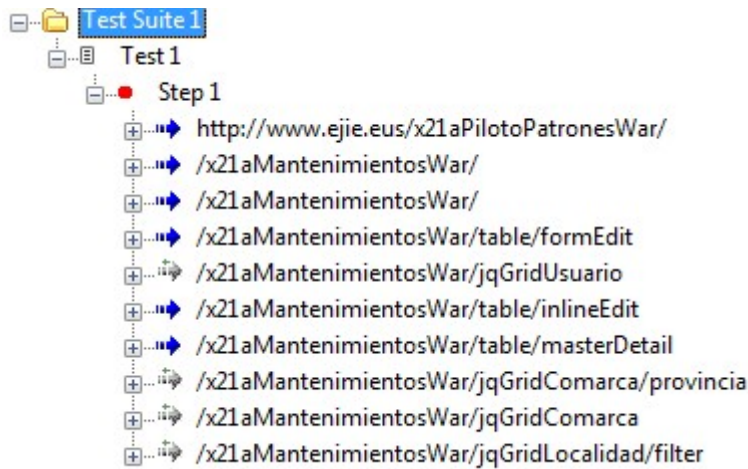
## 6 Anexo 1 – Ejemplo

Partiendo de un script de navegaciones generado con la herramienta Badboy sobre la aplicación Petstore (<http://www.ejje.eus/x21aPilotoPatronesWar/>) y realizada la exportación del script a interpretar por JMeter, vamos a crear su plan de pruebas correspondiente con la herramienta JMeter. Se ejecutarán pruebas de carga simulando peticiones de un número variable de usuarios.

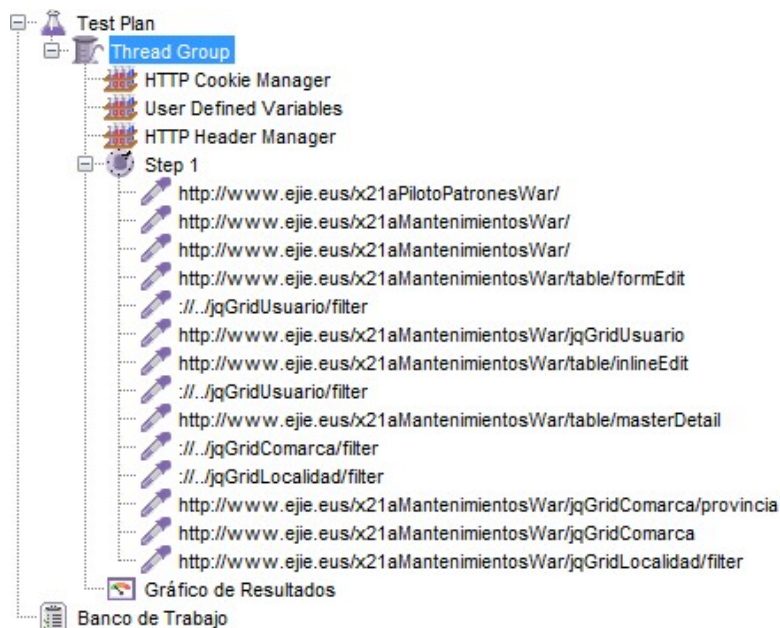
Se mostrarán los resultados obtenidos con ayuda de los siguientes “Listeners”: El “Gráfico de Resultados” y el “Summary Report” y se sacarán las conclusiones oportunas.

### 6.1 Resolución

En primer lugar obtendremos el script de navegación, realizada con Badboy, de la aplicación de ejemplo. La siguiente imagen da muestra del conjunto de navegaciones grabadas con Badboy para su posterior exportación como script de JMeter (JMeter se encargará de interpretar dicho script).



Exportando este script y abriéndolo con el JMeter (el script generado con Badboy y exportado para su posterior interpretación por JMeter) obtendremos el siguiente plan de pruebas:



Plan de pruebas cuyas propiedades se muestran en la siguiente figura:



0 / 10

### Plan de Pruebas

**Nombre:**

**Comentarios**

**Variables definidas por el Usuario**

Nombre:	Valor

☐ **Lanza cada Grupo de Hilos separadamente (i.e. lanza un grupo antes de lanzar el siguiente)**

☐ **Modo de Prueba Funcional**

Seleccione modo de prueba funcional solo si necesita archivar los datos recibidos del servidor para cada petición.  
Seleccionar esta opción impacta en el rendimiento considerablemente.

**Add directory or jar to classpath**           

Library

A su vez las propiedades del elemento "Thread Group" se reflejan en la siguiente imagen:

### Grupo de Hilos

Nombre:

Acción a tomar después de un error de Muestreador

☒ Continuar
 ☐ Parar Hilo
 ☐ Parar Test

---

Propiedades de Hilo

Número de Hilos

Periodo de Subida (en segundos):

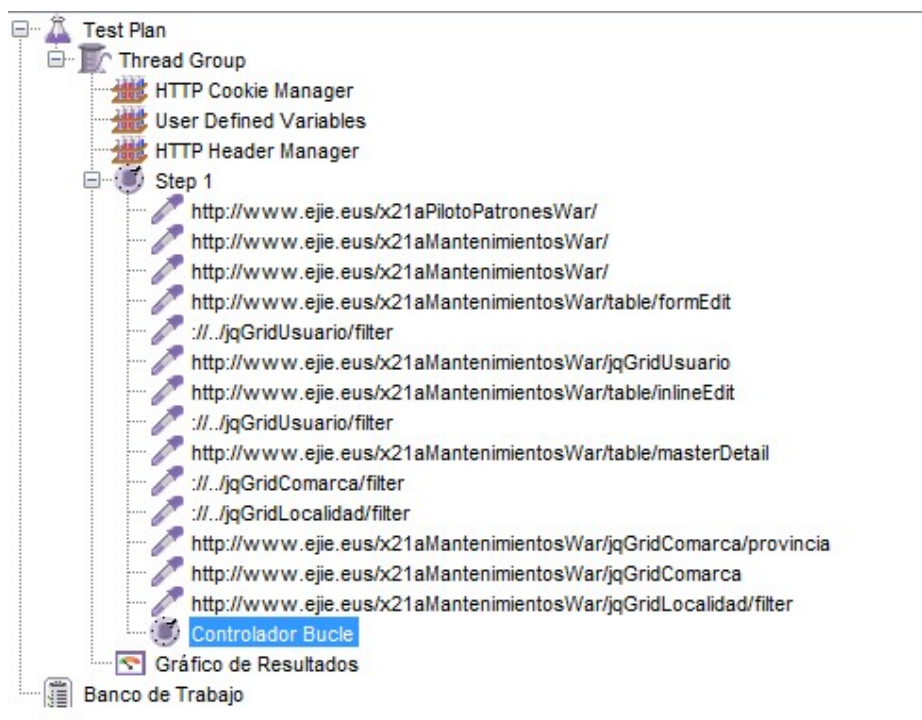
Contador del bucle: ☐ Sin fin

☐ Planificador

Una descripción sencilla de alguna de estas propiedades podrían ser la siguientes:

- **Número de hilos:** Número de usuarios a simular.
- **Periodo de subida (en segundos):** Tiempo que debiera llevarle a JMeter lanzar todos los hilos (si se seleccionan 10 hilos y el periodo de subida es de 100 segundos, entonces cada hilo comenzará 10 segundos después de que el hilo anterior haya sido lanzado).
- **Contador del bucle:** Número de veces a realizar el test.

Vemos que se ha obtenido un elemento de tipo “Contador Loop” generado durante la grabación del script por parte de Badboy. Se trata de un bucle que se ejecutará 10 veces, como muestra el valor 10 asociado a la propiedad “Contador del bucle” representado en la siguiente imagen:



The screenshot shows a JMeter Test Plan tree. Under a 'Thread Group', there are several components: 'HTTP Cookie Manager', 'User Defined Variables', 'HTTP Header Manager', 'Step 1' (containing a list of HTTP requests to ejje.eus), 'Controlador Bucle' (highlighted in blue), 'Gráfico de Resultados', and 'Banco de Trabajo'.

### Controlador Bucle

Nombre:

Comentarios

Contador del bucle: ☐ Sin fin

La siguiente figura muestra en detalle las diferentes peticiones HTTP grabadas en el script.



Podemos observar los valores de las propiedades de cada petición HTTP concreta: su nombre, el nombre del servidor al que se accede, el protocolo utilizado, etc.

**Petición HTTP**

Nombre:

Comentarios

Servidor Web

Nombre de Servidor o IP:

Petición HTTP

Implementación HTTP:  Protocolo:  Método:  Codificación del

Ruta:

☒ Redirigir Automáticamente ☐ Seguir Redirecciones ☒ Utilizar KeepAlive ☐ Usar \*multipa

Parameters **Body Data**

Enviar Parámetros Con la Pet

Nombre:

Detail Añadir Add from Clipboard

Enviar un archivo Con la Pet

Nombre de Archivo:

Añadir Navegar...

Servidor Proxy

Nombre de Servidor o IP:  Puerto:  Nombre

Embedded Resources from HTML Files

☐ Recuperar Todos los Recursos Empotrados de Archivos HTML ☐ Use concurrent pool. Size:

Dirección IP fuente:

IP/Hostna...

Tareas Opcionales

☐ Utilizar como Monitor ☐ ¿Gua

Situándonos, por ejemplo, sobre la última petición HTTP podremos observar los parámetros exportados en el script junto a la navegación:

**Petición HTTP**

Nombre:

Comentarios

Servidor Web

Nombre de Servidor o IP:

Petición HTTP

Implementación HTTP:  Protocolo:  Método:  Codificación del contenido:

Ruta:

☒ Redirigir Automáticamente ☐ Seguir Redirecciones ☒ Utilizar KeepAlive ☐ Usar 'multipart/form-data' pa

Parameters **Body Data**

Enviar Parámetros Con la Petición:

Nombre:	Valor
{ "filter": { "comarca": { "code": "11" } }, "_search": { "url": "../jqGridLocalidad/search", "a...	

Detail Añadir Add from Clipboard Borrar Up

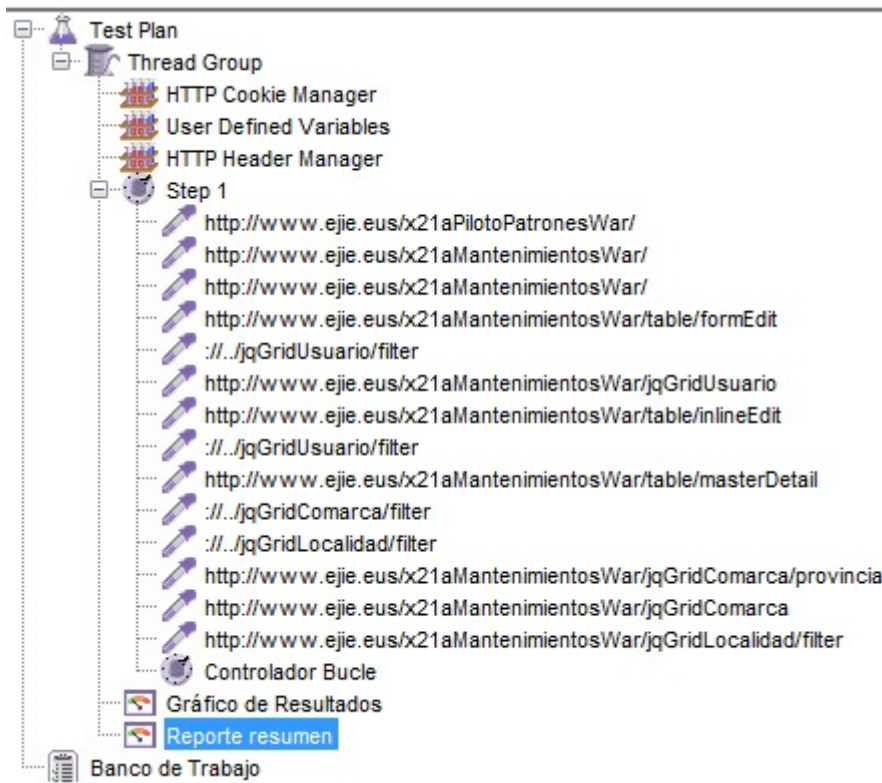
Enviar un archivo Con la Petición

Nombre de Archivo:

Una vez expuestos los elementos que componen el plan de pruebas es hora de introducir las variaciones y los elementos necesarios para realizar las pruebas adecuadas y obtener los datos estadísticos correspondientes. Para ello:

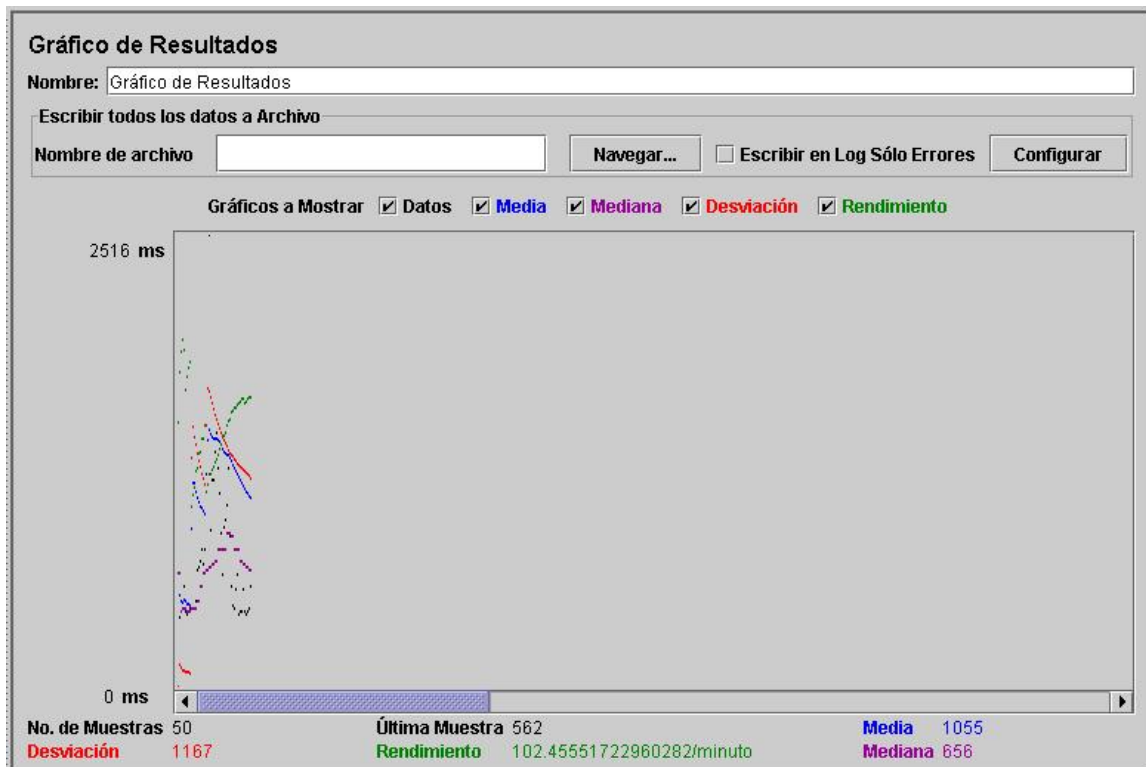
1. Se considerarán 10 hilos (simulación de 10 usuarios) y un periodo de subida de 30 segundos (3 segundos entre el lanzamiento de cada hilo). Por tanto se modificarán las propiedades "Número de Hilos" y "Periodo de subida (en segundos)" del elemento "Thread Group".
2. Se añadirá el elemento "Gráfico de Resultados". Para ello se considerará la opción del menú *Añadir > Listeners > Gráfico de Resultados*.
3. Se añadirá el elemento "Summary Report". Se considerará, por tanto, la opción del menú *Añadir > Listeners > Summary Report*.
4. Se modificará el nombre de las navegaciones para que los resultados de éstas se reflejen diferenciados en el "Summary Report".

Tras haber introducido las variaciones indicadas, el plan de pruebas presentará el siguiente aspecto:



### Simulación de 10 usuarios con un periodo de subida de 30 segundos

Una vez llegados a este punto será el momento de “Lanzar” las pruebas. Para ello haremos uso de la opción de menú *Lanzar > Arrancar*. Y obtendremos los resultados en el “Gráfico de Resultados”:



Una breve descripción de los elementos del gráfico puede ser la siguiente:

- **Datos:** muestra los valores actuales de los datos.
- **Media:** representa la Media.
- **Mediana:** dibuja la Mediana.
- **Desviación:** muestra la Desviación Estándar (una medida de la Variación).
- **Rendimiento:** representa el número de muestras por unidad de tiempo.

En la parte inferior de la ventana aparecen los valores actuales. “Última muestra” representa el tiempo transcurrido para la muestra en uso, valor mostrado en el gráfico como “Datos”.

El elemento “Summary Report” mostrará, a su vez, el siguiente aspecto:

**Summary Report**

Nombre: Summary Report

Escribir todos los datos a Archivo

Nombre de archivo  Navegar... ☐ Escribir en Log Sólo Errores Configurar

Label	# Muestras	Media	Mín	Máx	% Error	Rendimiento	Kb/sec	Avg. Bytes
http://www.	10	1718	422	5375	0,00%	21,9/min	3,12	8,56
http://www.	10	656	391	1219	0,00%	22,1/min	3,05	8,29
http://www.	10	694	422	1422	0,00%	22,1/min	3,11	8,45
http://www.	10	643	422	1078	0,00%	22,1/min	3,46	9,41
http://www.	10	1567	547	5235	0,00%	22,0/min	3,44	9,41
TOTAL	50	1055	391	5375	0,00%	1,7/sec	15,07	8,82

El “Summary Report” (Informe Resumen) crea una fila por cada petición (de diferente nombre) en el test (por eso se han renombrado cada una de las peticiones). Por cada una de estas filas se muestra la siguiente información:



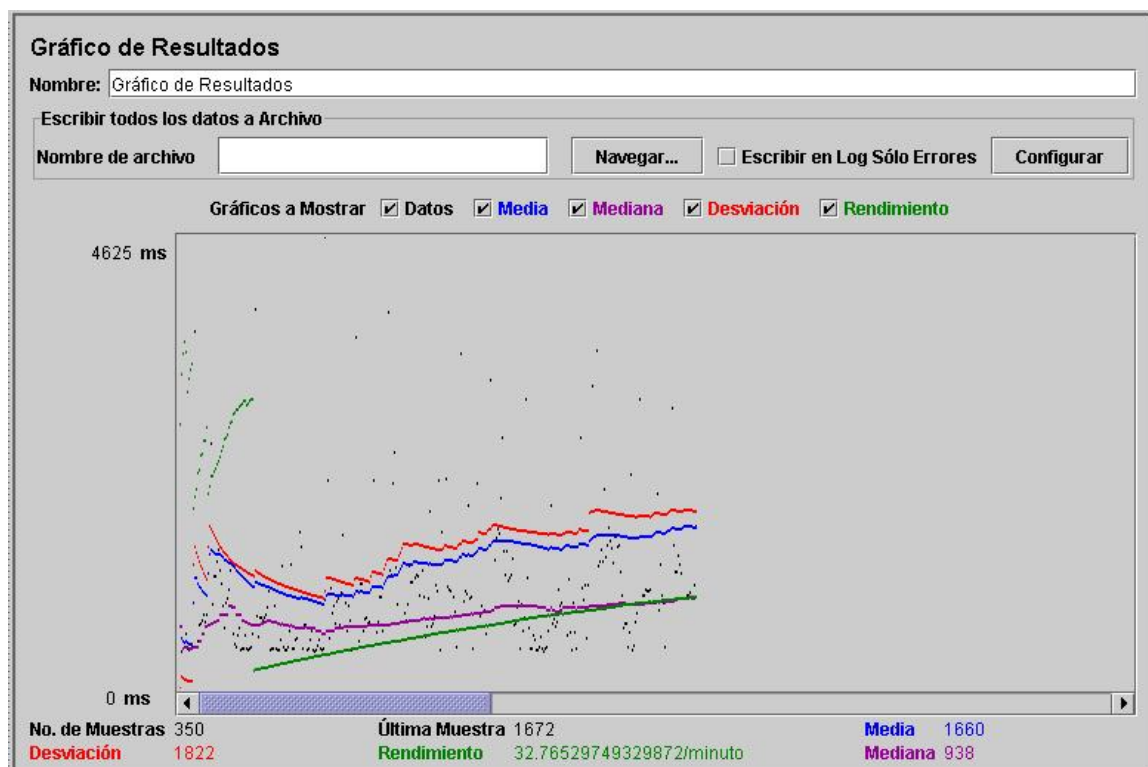
- **Label:** El nombre de la muestra (conjunto de muestras).
- **# Muestras:** El número de muestras para cada URL.
- **Media:** El tiempo medio transcurrido para un conjunto de resultados.
- **Mín:** El mínimo tiempo transcurrido para las muestras de la URL dada.
- **Máx:** El máximo tiempo transcurrido para las muestras de la URL dada.
- **Error %:** Porcentaje de las peticiones con errores.
- **Rendimiento:** Rendimiento medido en base a peticiones por segundo/minuto/hora.
- **Kb/sec:** Rendimiento medido en Kilobytes por segundo.
- **Avg. Bytes:** Tamaño medio de la respuesta de la muestra medido en bytes (erróneamente, en JMeter 2.2 muestra el valor en kB).

Podemos observar que las pruebas se han realizado sin errores. Esto se deduce de la columna representativa del tanto por ciento de errores para cada una de las peticiones asociadas a cada conjunto de muestras. El rendimiento nos muestra que para una simulación de 10 usuarios junto a un periodo de subida de 30 segundos el servidor es capaz de aceptar una media de 1,7 peticiones por segundo. La latencia (entendida como el tiempo de espera para la renderización de la página, el tiempo en obtener respuesta del servidor) para cada conjunto de pruebas no supera el valor de 2516 milisegundos (representado por el eje “y” de la gráfica).

### Simulación de 60 usuarios con un periodo de subida de 180 segundos

Si ahora realizamos la simulación con 60 usuarios considerando un periodo de subida de 180 segundos (de nuevo 3 segundos entre el lanzamiento de cada hilo) los resultados serán los siguientes, teniendo en cuenta que dichos resultados se irán solapando a los ya obtenidos en la simulación anterior.

El elemento “Gráfico de Resultados” mostrará un aspecto como el siguiente:



Podemos observar que la ejecución se ha detenido. El “Summary Report” ofrece el siguiente aspecto:

Summary Report									
Nombre: Summary Report									
Escribir todos los datos a Archivo									
Nombre de archivo				Navegar...		<input type="checkbox"/> Escribir en Log Sólo Errores		Configurar	
Label	# Muestras	Media	Mín	Máx	% Error	Rendimiento	Kb/sec	Avg. Bytes	
http://www.	70	2397	406	14938	0,00%	6,6/min	0,95	8,56	
http://www.	70	1248	390	6219	0,00%	6,6/min	0,91	8,27	
http://www.	70	1219	406	5890	0,00%	6,6/min	0,92	8,42	
http://www.	70	1634	421	6547	0,00%	6,6/min	1,03	9,38	
http://www..	70	1801	500	6782	0,00%	6,6/min	1,03	9,36	
TOTAL	350	1660	390	14938	0,00%	32,8/min	4,80	8,80	

En este caso volvemos a observar que las pruebas se han realizado sin errores. El rendimiento nos muestra que para una simulación de 60 usuarios junto a un periodo de subida de 180 segundos el servidor es capaz de aceptar una media de 32,8 peticiones por minuto. La latencia se ve aumentada hasta un valor de 4625 ms.

### Simulación de 120 usuarios con un periodo de subida de 360 segundos

Una vez obtenidos e interpretados estos resultados lanzaremos de nuevo simulando 120 usuarios:

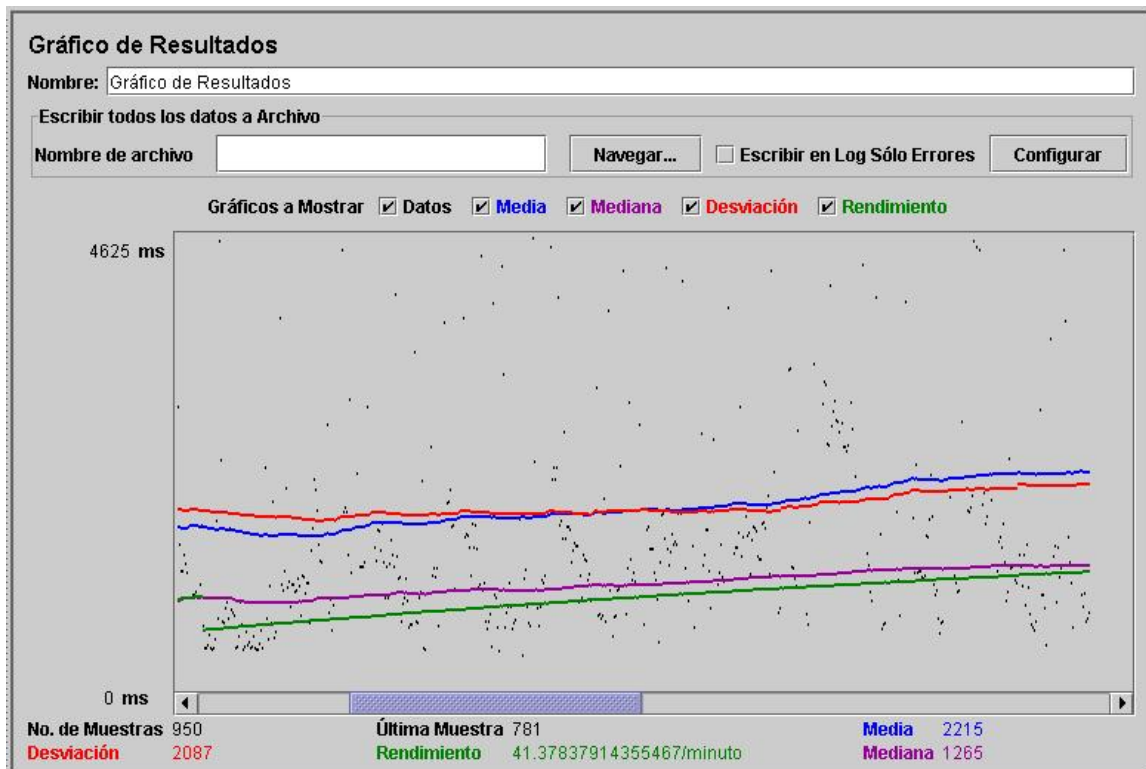
Se cambia el número de hilos a 120 con un periodo de subida de 360 segundos (por tanto se sigue manteniendo el tiempo entre hilos) y se estudian los resultados acumulados.

Podemos observar que la ejecución se ha detenido. El “Summary Report” ofrece el siguiente aspecto:

Summary Report									
Nombre: Summary Report									
Escribir todos los datos a Archivo									
Nombre de archivo				Navegar...		<input type="checkbox"/> Escribir en Log Sólo Errores		Configurar	
Label	# Muestras	Media	Mín	Máx	% Error	Rendimiento	Kb/sec	Avg. Bytes	
http://www.	190	2680	406	15906	0,00%	8,3/min	1,19	8,56	
http://www.	190	1792	343	7407	0,00%	8,3/min	1,14	8,27	
http://www.	190	2114	406	8266	0,00%	8,3/min	1,16	8,42	
http://www.	190	2152	421	9907	0,00%	8,3/min	1,30	9,37	
http://www.	190	2339	500	9328	0,00%	8,3/min	1,29	9,35	
TOTAL	950	2215	343	15906	0,00%	41,4/min	6,07	8,79	

El elemento “Gráfico de Resultados” mostrará un aspecto como el siguiente:





De nuevo, no hemos obtenido errores y puesto que se ha mantenido constante el tiempo transcurrido entre el lanzamiento de hilos el rendimiento apenas se ha visto afectado. La latencia se mantiene en el valor anterior de 4625 ms.

### Simulación de 120 usuarios con un periodo de subida de 120 segundos

En este momento se cambia el periodo de subida a 120 segundos manteniendo el número de hilos (1 segundo entre hilos).

Podemos observar que la ejecución se ha detenido. El "Summary Report" ofrece el siguiente aspecto:

**Summary Report**

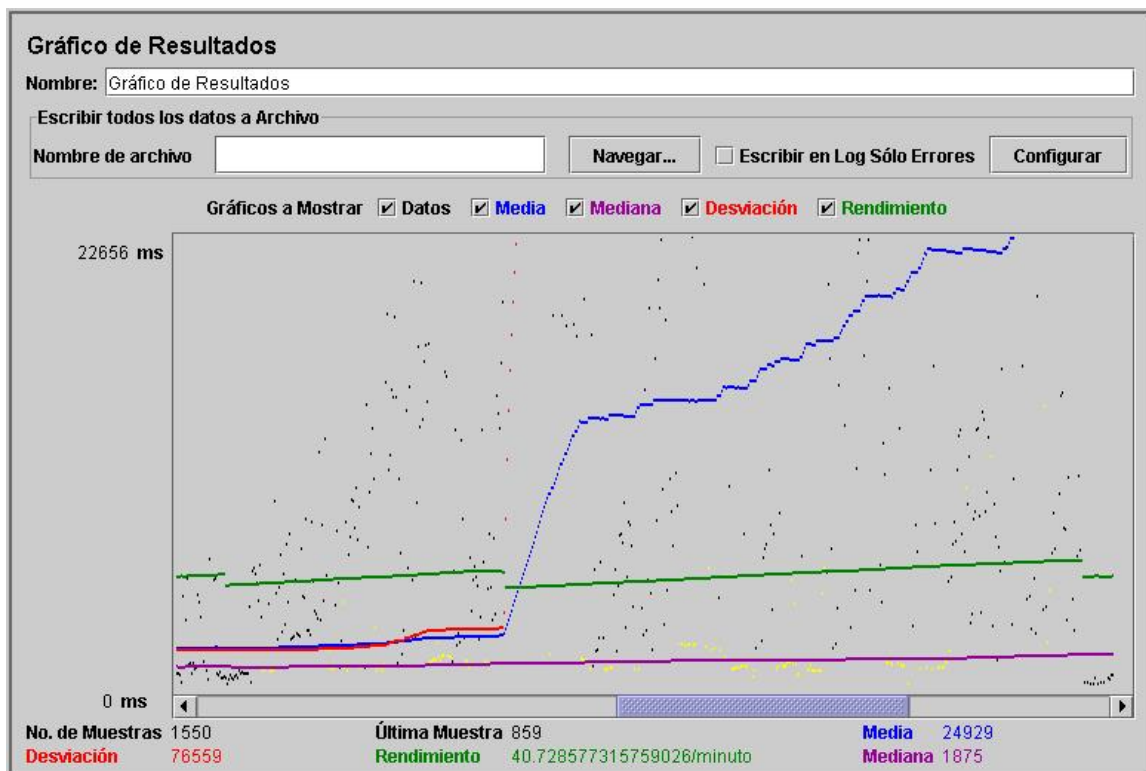
Nombre: Summary Report

Escribir todos los datos a Archivo

Nombre de archivo:  Navegar... ☐ Escribir en Log Sólo Errores Configurar

Label	# Muestras	Media	Mín	Máx	% Error	Rendimiento	Kb/sec	Avg. Bytes
http://www.	310	43421	406	304594	4,84%	9,6/min	1,31	8,17
http://www.	310	42275	343	389344	7,74%	9,5/min	1,19	7,55
http://www.	310	21566	406	391531	11,94%	8,2/min	1,00	7,33
http://www.	310	13707	421	370063	14,19%	8,2/min	1,06	7,78
http://www.	310	3677	390	41062	7,74%	8,2/min	1,08	7,92
TOTAL	1550	24929	343	391531	9,29%	40,7/min	5,26	7,75

El elemento "Gráfico de Resultados" mostrará un aspecto como el siguiente:



Observamos que comienzan a aparecer los primeros errores en un tanto por ciento considerable (9,29%) sobre el total de las peticiones realizadas por el global de las muestras. La latencia se ve aumentada hasta un valor de 22656 ms, lo cual se puede considerar como inaceptable.

### Simulación de 60 usuarios con un periodo de subida de 30 segundos

Por tanto se ejecuta un hilo cada medio segundo.

Los "listeners" nos ofrecen la siguiente información:

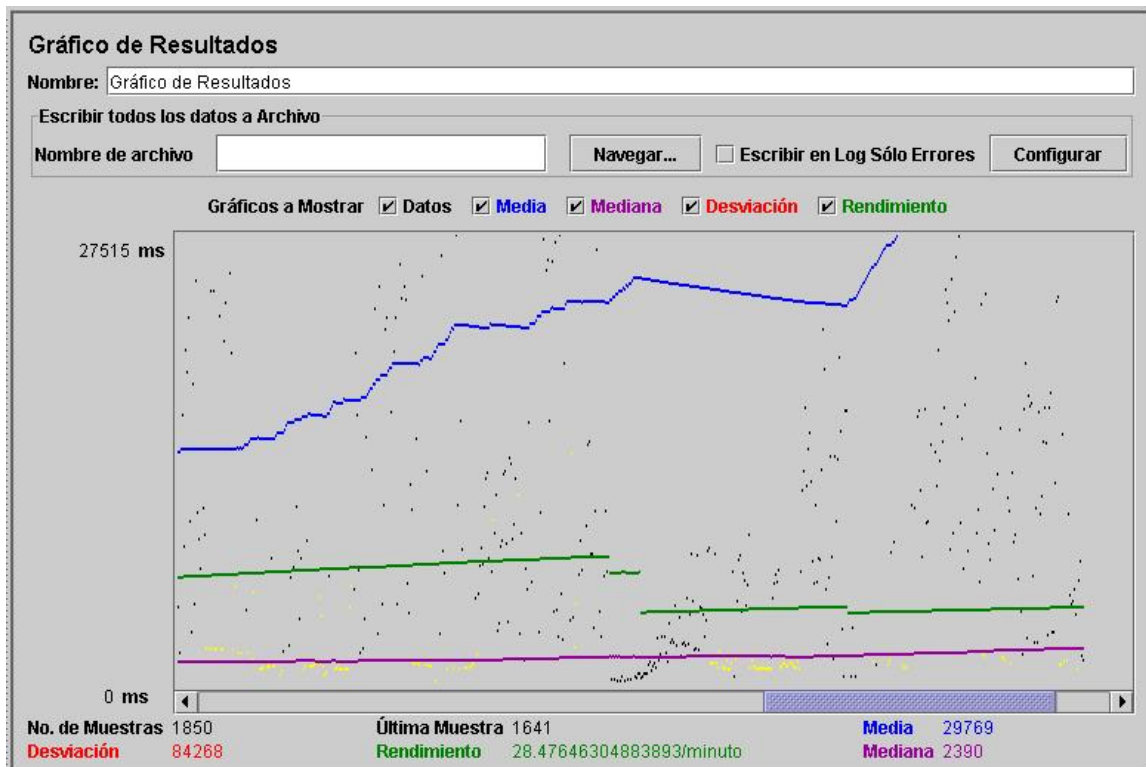
**Summary Report**

Nombre: Summary Report

Escribir todos los datos a Archivo

Nombre de archivo  Navegar... ☐ Escribir en Log Sólo Errores Configurar

Label	# Muestras	Media	Mín	Máx	% Error	Rendimiento	Kb/sec	Avg. Bytes
http://www.	370	37300	406	304594	8,38%	6,3/min	0,82	7,88
http://www.	370	62066	343	389344	11,08%	5,7/min	0,70	7,28
http://www.	370	28198	406	391531	13,78%	5,7/min	0,68	7,13
http://www.	370	17184	421	370063	14,32%	5,7/min	0,73	7,69
http://www.	370	4098	390	41062	9,19%	5,7/min	0,73	7,72
TOTAL	1850	29769	343	391531	11,35%	28,5/min	3,58	7,54



Observamos cómo cada vez se producen más errores y cómo disminuye el rendimiento. La latencia aumenta de nuevo hasta un valor de 27515 ms, considerada como inaceptable.

### Consideraciones

Habría que ajustar apropiadamente el volumen de estas pruebas puesto que tanto los resultados obtenidos como las conclusiones derivadas podrían estar condicionados por, por ejemplo, carencias de nuestra máquina en términos de memoria, etc.

Si las pruebas no pudieran llegar a su fin y hubiese que parar los hilos lanzados, esto podría deberse a dos motivos:

1. El servidor no puede trabajar con tantas peticiones.
2. La máquina local no soporta la creación de todas instancias (es en este caso en el que se puede hablar de las limitaciones de memoria).

En este último caso y para evitar este problema se podría actuar de alguna de las siguientes maneras:

1. Ejecutar JMeter por línea de comandos sin utilizar la interfaz gráfica de usuario.
2. Modificar la memoria RAM total que utilizará la máquina virtual de Java editando el fichero `jmeter/jmeter.bat` y añadiendo la sentencia

```
set HEAP=-Xms256m -Xmx256m
```

3. Como mejor solución realizar pruebas distribuidas (ver apartado 3.3).

## 7 Parametrización : JMeter – Script Badboy XLNets

Los pasos para parametrizar la herramienta en aplicaciones que necesiten autenticación en XLNets son los siguientes:

### 7.1 Crear el script de BadBoy

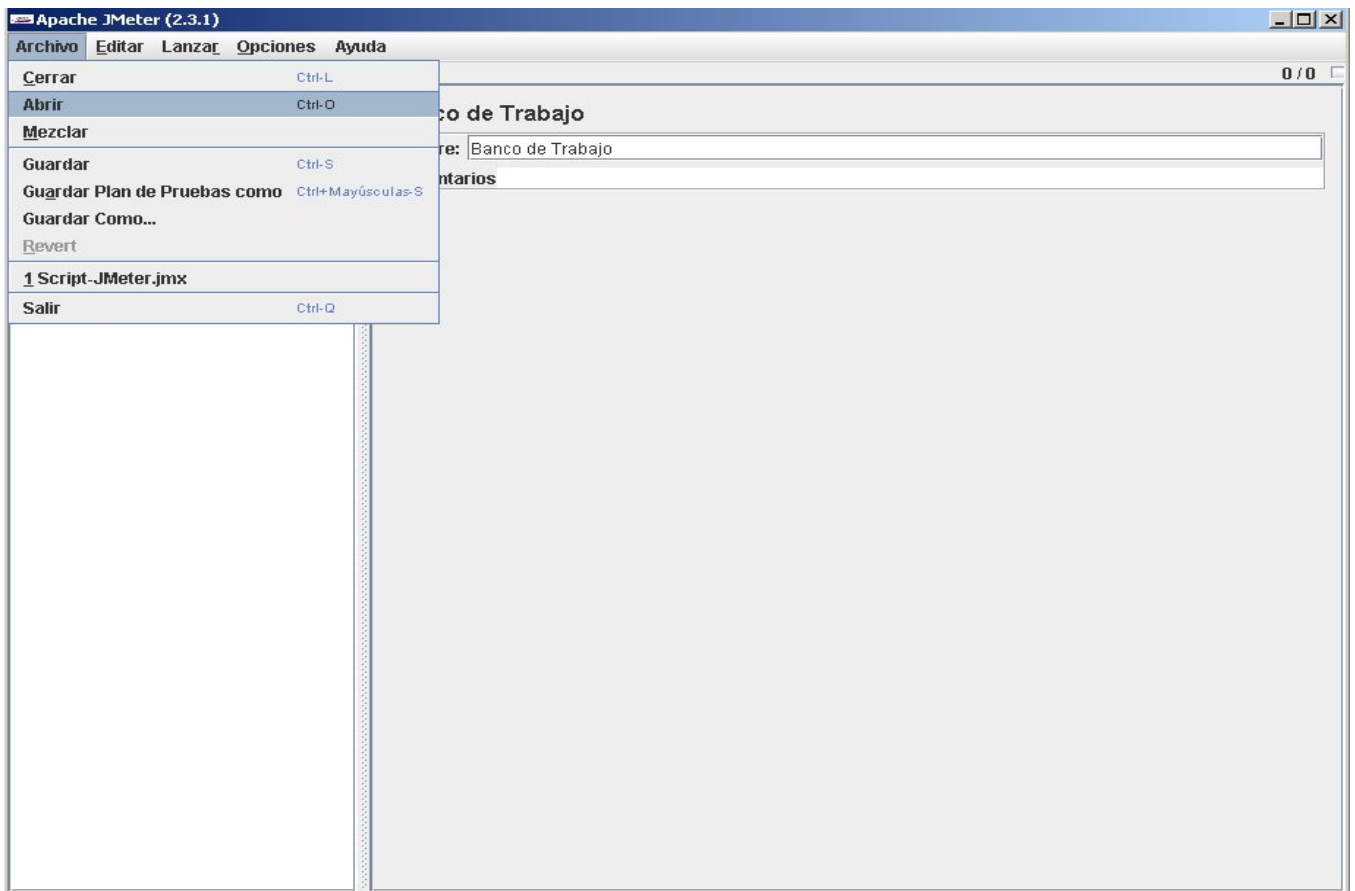
Ejecutamos el BadBoy. Inicialmente se ejecuta con el modo de grabación activado. Si nuestra aplicación usa la autenticación de usuario de XLNets, procederemos a desactivar el modo de grabación, antes de interactuar con nuestra aplicación.

Una vez que nos hemos autenticado en XLNets, y estamos en el punto de entrada de nuestra aplicación, procederemos a realizar la grabación, navegando por nuestra aplicación siguiendo los casos definidos.

Una vez finalizados los mismos, guardaremos el “script” con formato de BadBoy, por si más adelante quisiéramos añadir más navegación, y lo exportaremos también como “script” de JMeter.

### 7.2 Configurar las cookies en JMeter

Ejecutamos el programa JMeter, esta herramienta nos permitirá hacer las pruebas de carga siguiendo los “scripts” de navegación que hemos creado con “BadBoy”.



Una vez cargados los “scripts”, se deberán fijar las cookies necesarias para trabajar con XLNets, nos autenticamos en XINets con un usuario valido en la aplicación, abrimos (dobleclick) el candado de XINets y cogemos los datos necesarios para configurar nuestra navegación, el “UID de Sesión” y el “Dominio”.



**XL-NETs - Información de Sesión**

**Sesión:**

UID de Sesión: 1237299740125  
IP Host: 10.170.17.153 Dominio: D0\_Intranet11  
Usuario: T65H-3

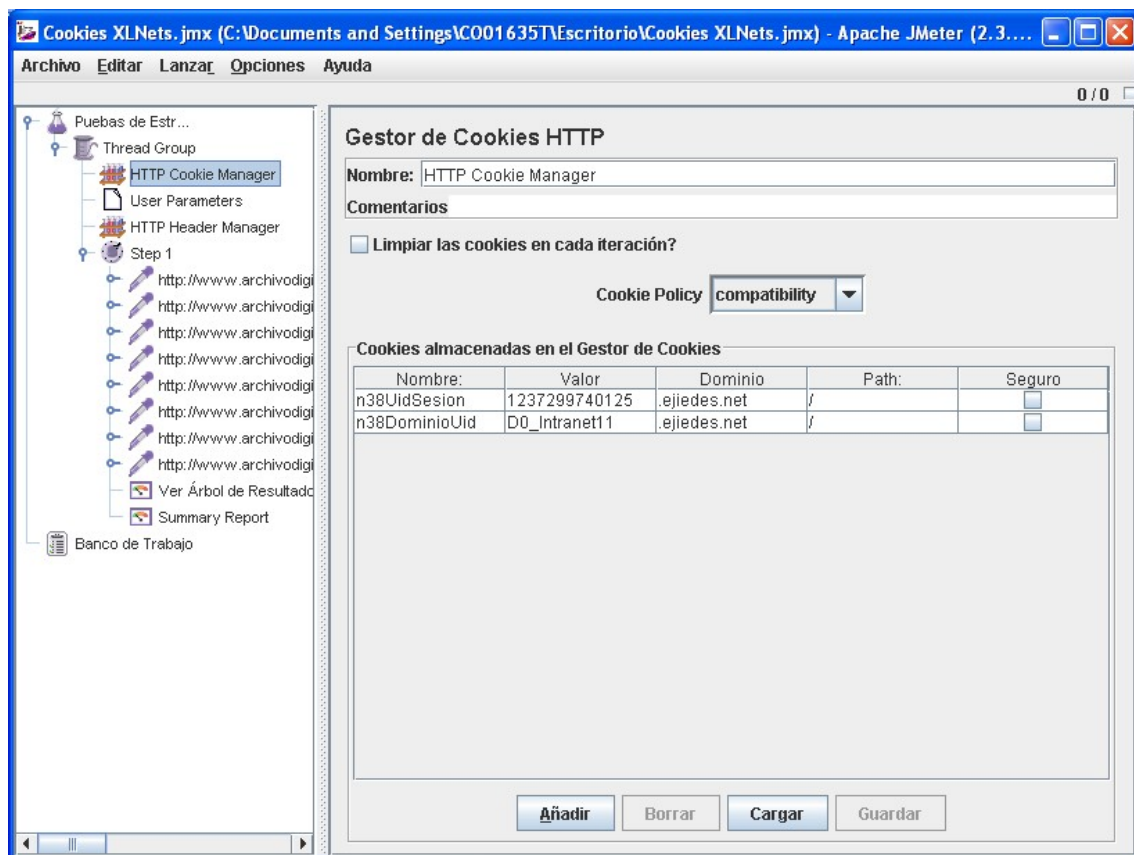
**Organización:**

Puesto: T65H-3  
Centro Orgánico: 16000  
Departamento: 500

**Validado en:**

.jakina.ejgvdns

Volvemos a la pantalla del JMeter, y en el Cookie Manager le damos al botón añadir, para fijar las siguientes cookies.



**Gestor de Cookies HTTP**

Nombre: HTTP Cookie Manager

Comentarios:

☐ Limpiar las cookies en cada iteración?

Cookie Policy: compatibility

**Cookies almacenadas en el Gestor de Cookies**

Nombre:	Valor	Dominio	Path:	Seguro
n38UidSesion	1237299740125	.ejjedes.net	/	<input type="checkbox"/>
n38DominioUid	D0_Intranet11	.ejjedes.net	/	<input type="checkbox"/>

En el caso del ejemplo anterior, los valores a rellenar para cada cookie serán por tanto:

- **n38UIdSesion-valor:** 1237299740125 (obtenido del candado)
- **n38DominioUId-valor:** D0\_Intranet11 (obtenido del candado)

El resto de propiedades se rellenaran para las dos de la siguiente manera:

- **Dominio:** .ejjedes.net (depende del dominio de la aplicación, no olvidar el punto inicial)
- **Path:** "/" (siempre tendrá este valor)

El nombre de las cookies no variará nunca, lo que puede variar son los valores de las cookies "n38UIdSesion" y "n38DominioUId" y solo si ejecutamos el plan de pruebas en sitios diferentes.