JMeter

Víctor Herrero Cazurro

Contenido

1. Introducción
2. Instalación
3. Plan de Pruebas
4. Banco de Trabajo
5. Jmeter Maven Plugin

Introducción

- Herramienta independiente para realizar pruebas funcionales y de stress sobre un servidor Web.

http://jakarta.apache.org/jmeter/

Introducción

- JMeter nos ayudara a detectar
  - Cuellos de botella en el sistema.
  - Fallos en aplicaciones.
  - Peticiones que es capaz de absorber el servidor.
  - Simulación de un uso cotidiano de una aplicación.
  - Simulación de estrés de una aplicación.
  - …

Introducción

- Que no puede hacer
  - JMeter simplifica la generación de los planes de Test, pero no puede generarlos por si mismo.
  - Se precisa de tiempo para generar planes de Test eficientes.

Introducción

- Al acceder a JMeter se ve que la aplicación esta dividida en dos partes.
  - **Plan de Pruebas**. Donde desarrollaremos nuestros planes de pruebas.
  - **Banco de Trabajo**. Donde tendremos las herramientas y operaciones a utilizar en nuestro plan de pruebas.
  - Podremos mover contenidos del Banco de Trabajo a los Planes de Prueba para su uso.

Instalación

- La descarga se realiza desde la siguiente URL.

```
http://jakarta.apache.org/site/downloads/downloads_jmeter.cgi
```

- Necesaria JVM. Compatibilidad 1.5x en adelante.
- Para asignar memoria a JMeter, emplearemos la variable de entorno JVM_ARGS.

```
JVM_ARGS=-Xms256m -Xmx512m
```

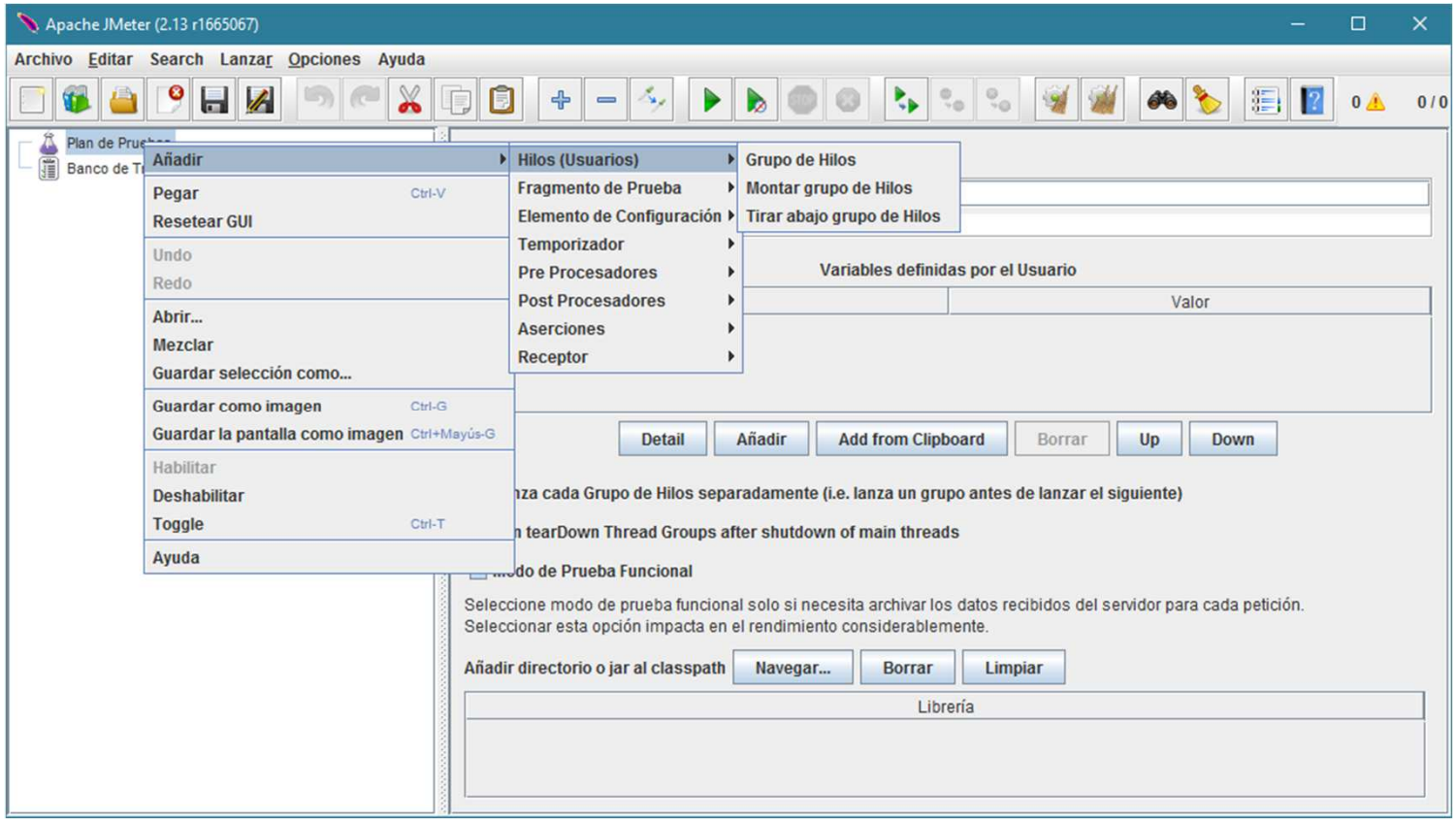
- Para incluir un jar en nuestros test, por ejemplo el driver de una base de datos, se incluirá en la carpeta lib.

Plan de Pruebas

- Nuestro Plan de Pruebas estará compuesto por elementos de las siguientes tipologías.
  - **Grupo de Hilos**.
  - **Procesadores**.
  - **Receptores o Listeners**.
  - **Controladores**.

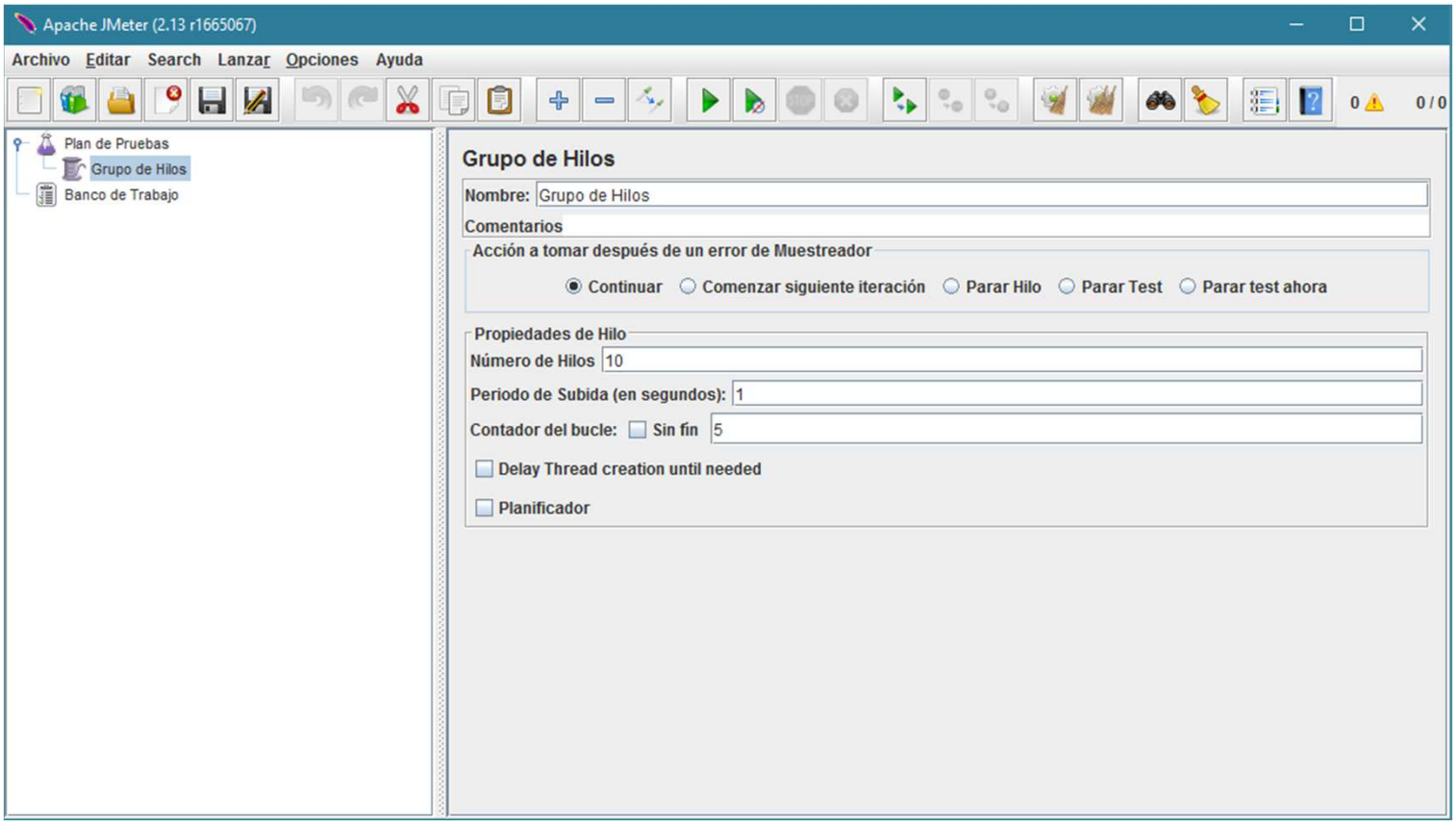
Grupo de Hilos

- Simulara al numero de usuarios.



Grupo de Hilos

- Permite definir usuarios simultáneos y peticiones de cada usuario.



Procesadores

- Que se dividen en
  - **Pre-Procesadores**. Modifican la petición antes de ejecutarse.
  - **Post-Procesadores**. Modifican la petición después de ejecutarse.

Controladores

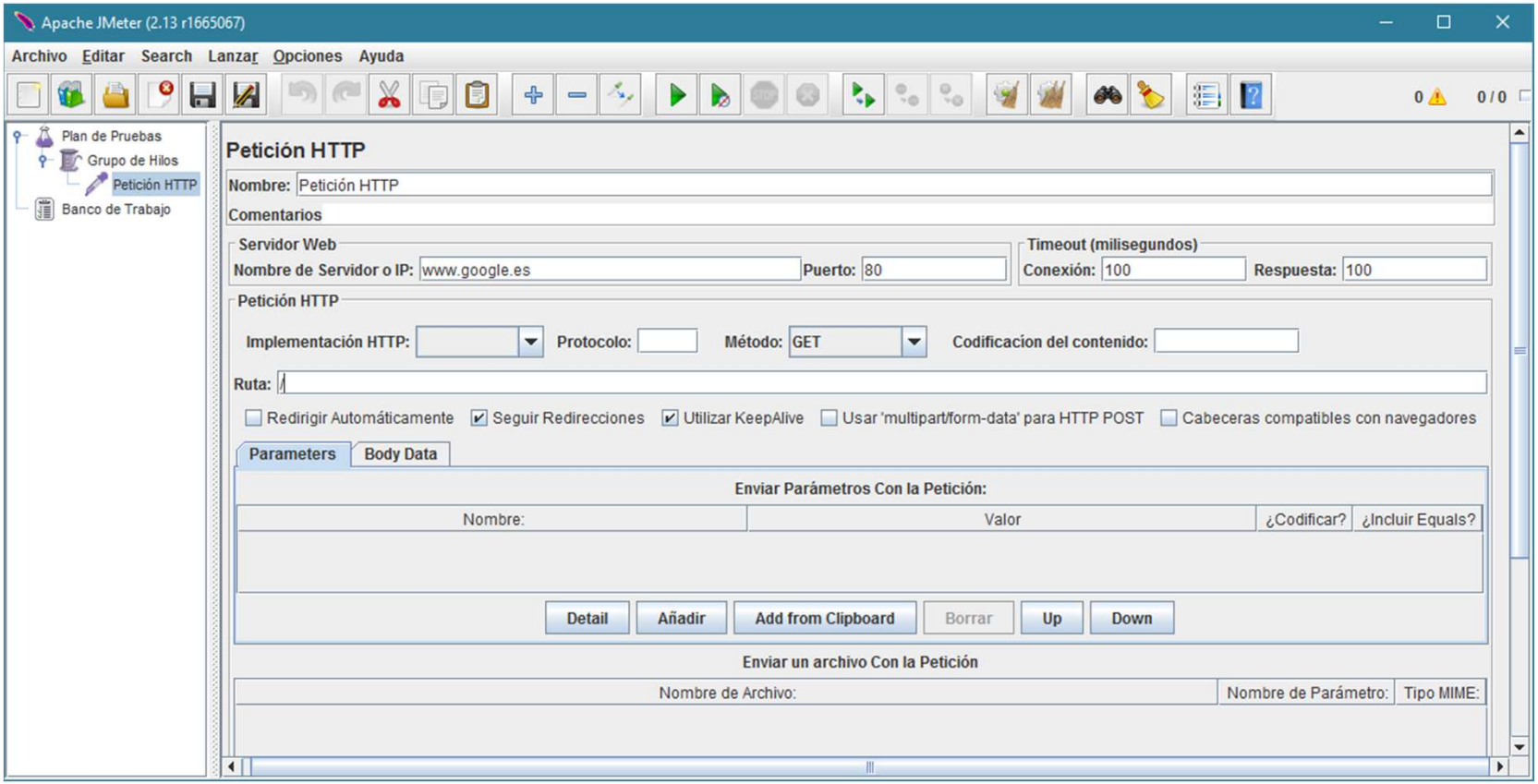
- Que a su vez se dividen en
  - **Muestreadores**. Indican las acciones que JMeter puede hacer.
  - **Aserciones**. Comprobación de resultados esperados.
  - **Elementos de configuración**. Trabaja en conjunto con los Muestreadores para modificarlos.
  - **Controladores lógicos**. Modifican la lógica de lo que se debe hacer (Toma de decisiones).
  - **Temporizadores**. Incluye pausas en el test, para simular la realidad.

Controladores: Muestreadores

- Para configurar que acciones debe realizar nuestro Test de JMeter, se deben añadir **Muestreadores**.
- Tenemos entre otros los siguientes tipos:
  - **Petición HTTP**. Nos permite realizar una petición HTTP, indicando protocolo, método, parámetros, …
  - **Petición JDBC**. Nos permite ejecutar una consulta sobre la Base de Datos.
  - **Petición WebServices (SOAP)**. Permite realizar peticiones SOAP a un servicio web, empleando el WSDL.
  - …

Controladores: Muestreadores

- Ejemplo de Muestreador de Petición Http

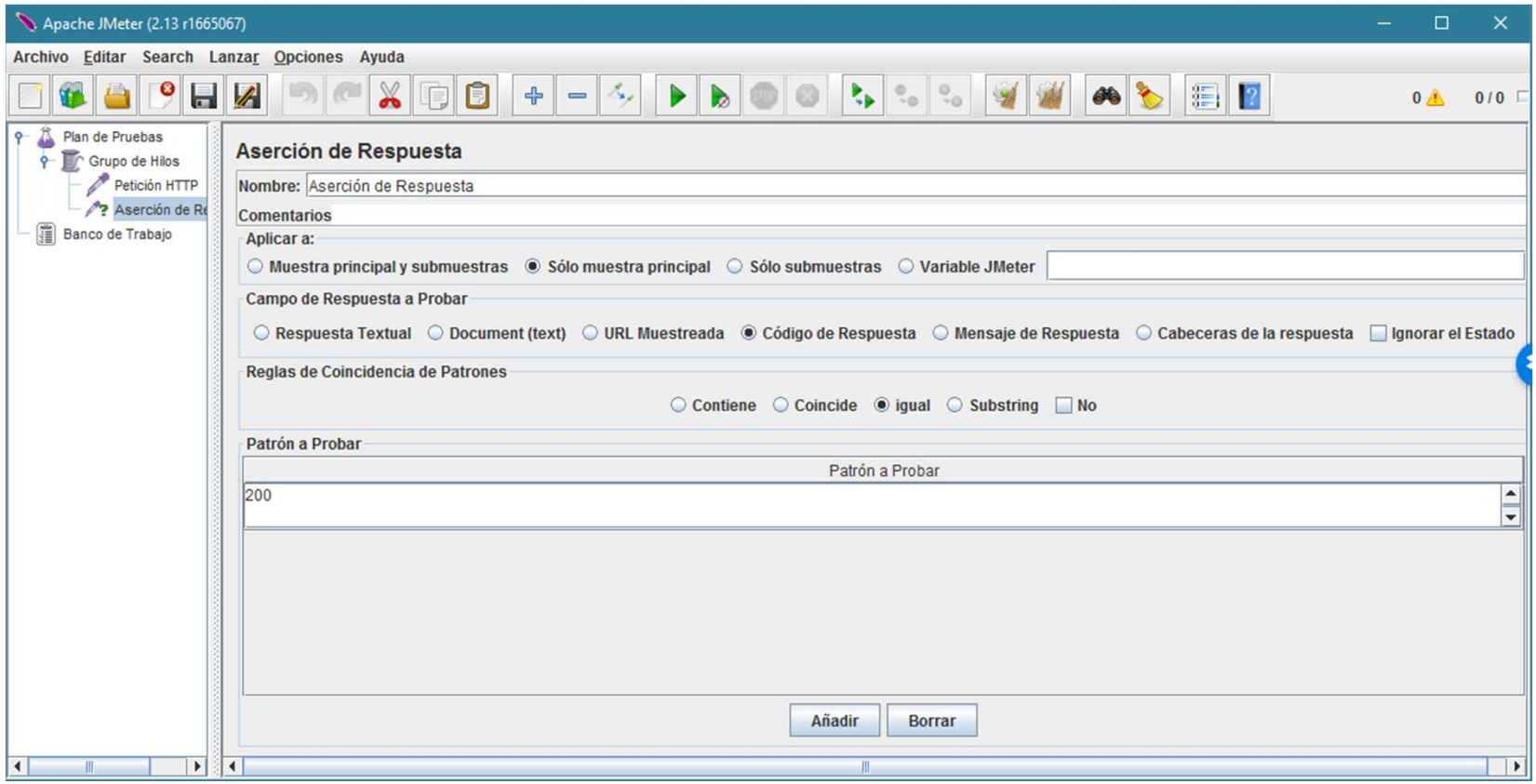


Controladores: Aserciones

- Para configurar que aserciones debe realizar nuestro Test, JMeter nos proporciona los siguientes comandos:
  - **Aserción de respuesta**. Nos permite comprobar si alguno de los campos de la respuesta coincide con un determinado patrón, OJO!! es la respuesta no el HTML.
  - **Aserción de esquema XML**. Valida que la respuesta cumple el esquema indicado mediante un fichero XSD.
  - **Aserción XML**. Comprueba que el resultado es un XML bien construido.
  - **Aserción de Tamaño**. Nos permite comprobar si alguno de los campos de la respuesta tiene un tamaño determinado.
  - …

Controladores: Aserciones

- Ejemplo de Aserción de Respuesta basada en el código de respuesta HTTP



Receptores o Listeners

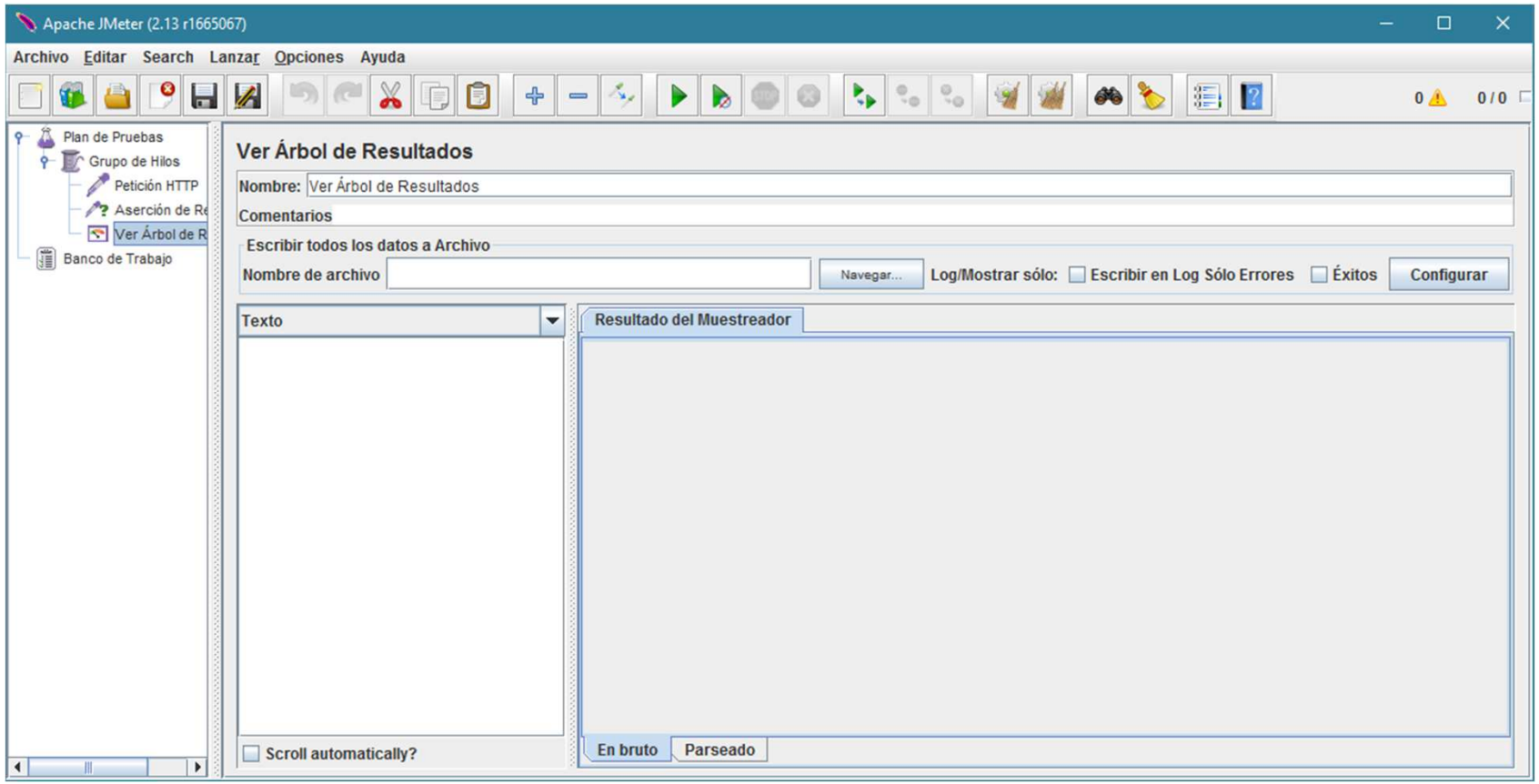
- Nos mostraran los resultados de las peticiones en distintos formatos.
- Para mostrar los resultados se debe añadir un **Listener** al plan de pruebas.

Receptores o Listeners

- Tenemos entre otros los siguientes tipos:
  - **Árbol de resultados**. Para cada petición muestra la respuesta HTTP, la petición y los datos HTML devueltos.
  - **Informe agregado**. Muestra un resumen de los resultados
  - **Gráfico de resultados**. Muestra un gráfico de rendimiento
  - …

Receptores o Listeners

- Ejemplo de Receptor de Árbol de resultados



Ejemplo: Enunciado

- El primer plan de pruebas que vamos a crear, deberá acceder a un servidor Tomcat, y comprobar lo siguiente.
  - Que se conecta (Código de Respuesta 200).
  - Que el HTML devuelto contiene el siguiente Texto “Apache Tomcat”.
- Hasta aquí las pruebas de funcionalidad, pero vamos a querer también comprobar como se comporta el servidor ante una concurrencia de usuarios (rendimiento), por lo tanto vamos a simular que se conectan 50 usuarios a la vez, y cada uno de ellos realiza 5 peticiones sobre la misma página.
- Añadiremos un Listener, para comprobar los resultados, por ejemplo el “Resumen del Reporte”.

Ejemplo

- Para crear el primer Plan de Pruebas, vamos a definir un Grupo de Hilos para lanzar llamadas concurrentes.
- Se puede indicar
  - Nombre del grupo.
  - Número de hilos.
  - Periodo de subida (Tiempo en lazar todos los hilos).
  - Número de veces que se lanza una petición al servidor.
- De esta forma vamos a establecer el numero de veces que se realiza nuestra prueba, y como se realizan esas peticiones, si cada vez una, o varias a la vez.

Ejemplo

- Posteriormente añadiremos al grupo de hilos las acciones que queremos llevar a cabo.
- En este caso, es una petición HTTP, por lo que necesitaremos un “Muestreador”, en este caso “Petición HTTP”, en el que podemos indicar.
  - Máquina (nombre o IP)
  - Puerto (80)
  - Protocolo (http)
  - Método de petición (GET/POST)
  - Posibles parámetros para la petición
  - …
- Antes podemos incluir un “Elemento de Configuración”, para definir una configuración por defecto.

Ejemplo

- Una vez establecidas las acciones, es turno de las aserciones. Como lo que queremos comprobar es la respuesta a nuestra petición, incluimos unas aserciones asociadas a la petición, usaremos “Aserción de Respuesta”. En ella podemos indicar.
  - Campo a comprobar.
  - Tipo de comprobación.
  - Valor esperado.
- Por ultimo añadimos el visor de los resultados, en este caso “Reporte Resumen”.

Banco de Trabajo

- Como decíamos el Banco de Trabajo será el lugar donde tengamos nuestras herramientas que nos ayudaran a configurar nuestro Plan de Pruebas, una de las herramientas mas interesantes que tendremos será el **Servidor Proxy HTTP**.
- Esta herramienta nos permitirá obtener los pasos seguidos en una navegación, es decir es capaz de traducirnos a acciones de JMeter, los pasos que hacemos en una navegación, para posteriormente poder grabarlos como macro de JMeter.

Servidor Proxy HTTP

- Para utilizar esta herramienta, debemos seguir los siguientes pasos:
  - Creamos en el Banco de Trabajo un elemento Servidor Proxy HTTP.
  - Configuramos el controlador objetivo, es decir donde queremos que vaya poniendo los pasos que se vayan a seguir en la navegación.
  - Establecemos los patrones a incluir o excluir en la navegación, es posible que no interese que se almacenen imágenes, javascript, …
  - Arrancamos el Proxy.
  - Configuramos el navegador a utilizar, para que pase a través de este proxy.
  - Realizamos la navegación.
  - Paramos el Proxy.

Jmeter Maven Plugin

- Existe un plugin de Maven para lanzar test de Jmeter

```
<plugin>
        <groupId>com.lazerycode.jmeter</groupId>
        <artifactId>jmeter-maven-plugin</artifactId>
        <version>1.4.1</version>
        <executions>
                <execution>
                        <id>jmeter-tests</id>
                        <phase>verify</phase>
                        <goals>
                                <goal>jmeter</goal>
                        </goals>
                </execution>
        </executions>
</plugin>
```

Jmeter Maven Plugin

- Este plugin, correra todos los ficheros jmx, que estén en el directorio.

```
src/test/jmeter
```

- Dado que se ha integrado con el ciclo de ida del proyecto, se ejecutara siempre que se pase por verify.

```
mvn verify
```

- O ejecutando directamente el goal

```
mvn jmeter:jmeter
```

Jmeter Maven Plugin

- Si se añade a Jenkins en plugin Performance, se consigue integrar los resultados de Jmeter en Jenkins

Ejercicio

- Dada una base de datos, con una tabla “usuarios”, realizar un plan de pruebas para comprobar que nuestra base de datos acepta una conexión concurrente de 3 usuarios, que realizan una serie de operaciones 10 veces.
- Las operaciones serían:
  - Consultar cuantos libros hay en la tabla de libro.
- Se debe obtener el resultado de cada una de las peticiones, así como un grafico de rendimiento.

Ejercicio

- Crear un Plan de Pruebas, para comprobar el rendimiento de la aplicación “Servidor”, comprobando por ejemplo que en el HTML recibido una vez que se envía el formulario, están presentes los textos enviados por el formulario.