Published on *Marco de Desarrollo de la Junta de Andalucía* (http://madeja.i-administracion.junta-andalucia.es/servicios/madeja)

Construcción de Planes de Prueba con JMeter

- **Área:** Verificación de Entrega Software
- **Carácter del recurso:** Recomendado

Código: RECU-0391
Tipo de recurso: Manual

Descripción

JMeter es una herramienta de testing cuyas funcionalidades se pueden resumir en tres:

- Diseñar un testplan, esto es, generar un fichero .jmx. Este es el objeto de este documento
- Ejecutar un testplan
- Ver de distintas formas los resultados de la ejecución de un testplan (vía *listeners*)

Para diseñar un testplan, JMeter dispone de una interfaz GUI a modo de diseñador, en la que el tester puede ir agregando componentes de manera visual, y ejecutar los componentes agregados, viendo el resultado. Una vez finalizado el diseño del testplan, la herramienta permite grabar este como un fichero .jmx.

La propia herramienta permite ejecutar un fichero .jmx previamente generado, vía línea de comandos o vía la propia interfaz GUI. La ejecución de un fichero .jmx realiza peticiones contra la aplicación objetivo a testear (peticiones del tipo que se hayan especificado al generar el fichero .jmx, JMeter dispone de la posibilidad de generar muchos tipos de peticiones: HTTP, FTP, LDAP, ...). Para cada petición ejecutada, JMeter recopila ciertos datos. Además, en el fichero .jmx se puede especificar que número de usuarios de cada tipo ejecuta las peticiones contra la aplicación, es decir, el .jmx simula una o mas comunidades de usuarios (roles) trabajando contra la aplicación objetivo.

Los datos generados por la herramienta para cada petición se procesan o bien con un tipo de componente que proporciona la interfaz GUI llamados *listeners*, o bien con herramientas externas. Los listeners permiten ver los resultados de una o mas ejecuciones de múltiples maneras (cada listener de una manera).

Este manual explica los conceptos necesarios para construir testplans (ficheros .jmx) configurables y mantenibles con JMeter. En particular, se explica el concepto de componente y tipo de componente de un testplan, el orden en que estos se asocian a los samplers, el ámbito de los diferentes tipos de componentes, los métodos para grabar un testplan, y como construir testplan configurables y mantenibles mediante el uso de variables, propiedades y funciones.

Está orientado principalmente a técnicos y testers involucrados en la construcción de ficheros .jmx de JMeter o a técnicos que deseen iniciarse en el uso de la herramienta para implementarlos.

Los lectores con conocimientos previos en JMeter que sólo estén interesados en la ejecución de testplans, no en su construcción, deberían consultar esta otra referencia de MADEJA, en lugar de este documento: Ejecución de un testplan con JMeter. En cualquier caso, para estos lectores también aplican los puntos de este documento relativos a los tipos de componentes de JMeter, ámbito y orden de ejecución de los mismos.

En cualquier caso, tanto este documento como la referencia mencionada más arriba, asumen cierto nivel de conocimientos previos en JMeter. Al menos los conceptos básicos sobre como funciona la herramienta, y como se usa en líneas generales. Los usuarios sin conocimientos previos que necesiten adquirir estos conocimientos sobre JMeter, pueden consultar esta otra referncia de MADEJA: Introducción a JMeter: Conceptos Básicos.

Ventajas e inconvenientes

Ver la referencia Introducción a JMeter / Ventajas e inconvenientes.

Prerequisitos e incompatibilidades de la tecnología

Ver la referencia Introducción a JMeter / Prerequisitos e incompatibilidades de la tecnología

Relación con otros componentes y subsistemas

Ver la referencia Introducción a JMeter / Relación con otros componentes y subsistemas

Modo de empleo

Para la instalación de la herramienta y configuración de idioma, consultar la referencia Introducción a JMeter / Modo de empleo /

Instalación, configuración, y acceso a la documentación

NOTA:

Por qué es importante la configuración del idioma (Inglés) en la herramienta: la interfaz GUI dispone de un sistema de ayuda online que muestra la documentación de referencia de un componente (opción Help del menú emergente del componente en el árbol del testplan). Este sistema de ayuda online obtiene la documentación de ayuda accediendo, vía http a la referencia de componentes sita en la web del proyecto JMeter, realizando una búsqueda por el nombre del componente en INglés. Si la interfaz GUI se configura en Español, el sistema de ayuda obline no encontrará ninguna concidencia y no funcionará correctamente.
Este detalle es importante sobre todo para aquellos que se incian en la construcción de testplans, pues necesitan consultar frecuentemente la referencia de los componentes de JMeter.

Modos de ejecución de JMeter

Hay dos formas de ejecutar JMeter según que queramos que se muestre o no la interfaz GUI:

- Desde línea de comando SIN mostrar la interfaz GUI (especificar la opción -n en la línea de comando).
- Mostrando la interfaz GUI: no especificar la opción -n al arrancar JMeter.
  La mayoría de las restantes opciones de la línea de comando son aplicables a la ejecución de JMeter tanto si se utilizan con el modificador -n como sin él.

Para diseñar un testplan, utilizamos la interfaz GUI, una de cuyas funcionalidades es además de ejecutar testplans, proporcionar un entorno para diseñarlos. Por tanto, para diseñar un testplan arrancaremos JMeter sin el modificador -n, por ejmeplo:

```
$ ${JMETER_HOME}/bin/jmeter
```

Los puntos que siguen asumen que el lector conoce las interfaz GUI de JMeter, si no es el caso, consultar la referencia Introducción a JMeter: Conceptos Básicos / Artefactos software.

Componentes y tipos de componentes de un testplan

Un testplan (fichero .jmx) es una JERARQUÍA de componentes en forma de árbol. Puede verse abriendo un fichero .jmx en la interfaz GUI, en el frame de la izquierda (en el directorio de la instalacion ${JMETER_HOME}/printable_docs/demos/ hay varios .jmx de ejemplo).

Cada nodo del árbol es un componente. A su vez, un componente es una instancia de un tipo de componente en la que quizás se han configurado algunas de sus propiedades (en el panel de control de la derecha).

La tabla siguiente explica para que sirve cada uno de los tipos de componentes que existen en JMeter:

Tipo de componente	Uso
Testplan	Es el tipo de componente que representa la raíz del árbol. En todo testplan existe uno y sólo un componente de este tipo
Thread Group	Representa un grupo de usuarios. En JMeter cada thread es un usuario virtual. Este tipo de componente permite representar grupos de usuarios. Cada grupo de usuarios del testplan representa un rol o perfil. Todos los threads de un mismo thread group (i.e. todos los usuarios de un grupo) realizan la misma secuencia de acciones, representada por los samplers que agrupa el thread group.
Controllers: Sampler, Logic Controler	Son los únicos componentes del testplan que hacen algo: los samplers realizan peticiones contra la aplicación, y los logic controlers establecen el orden en que se ejecutan los samplers que agrupan. El resto de componentes (Config Element, Assertion, ...) "matizan" la forma en que se ejecutan los samplers, pero no varían sustancialmente su comportamiento
Config Element	Establecen propiedades de configuración que se aplican a los samplers a los que afectan
Assertion	Comprueban condiciones que aplican a las peticiones que los samplers a los que afectan realizan contra la aplicación
Listener	Recopilan datos de las peticiones que realizan los samplers a los que afectan
Timer	Añaden tiempo extra a la ejecución de las peticiones que realizan contra la aplicación los samplers a los que afectan
Pre-Processor element	Realizan acciones o establecen configuraciones previa a la ejecución de los samplers a los que afectan
Post-Processor element	Realizan acciones o establecen configuraciones posteriormente a la ejecución de los samplers a los que afectan

El apartado 4. Elements of a Test Plan del manual de usuario de JMeter explica en detalle cada uno de los tipos de componentes.

El apartado 18. Component Reference del manual de usuario de JMeter contiene la referencia de todos los tipos de componentes que existen en JMeter.

Ambito y orden de ejecución de componentes de un testplan

Cada componente, según su tipo, es un elemento de Orden (O = Ordered) o Jerárquico (H = Hierarchy).

Los tipos de componentes son (entre paréntesis indicamos si es un elemento de orden o jerárquico):

- Testplan
- Thread Group
- Controllers: Sampler, Logic Controler (O)
- Config Element (H)
- Assertion (H)
- Listener (H)
- Timer (H)
- Pre-Processor element (H)
- Post-Processor element (H)

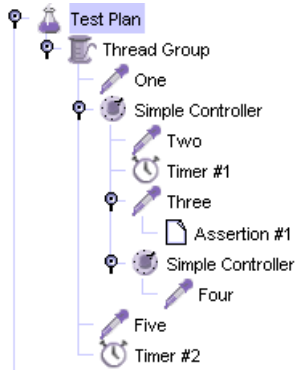
El ámbito establece a que elementos de tipo O (samplers y logic controlers) afecta un elemento de tipo H.

El orden de ejecución establece en que orden se ejecutan los elementos de un test plan según su tipo y ciertas reglas de asociación entre ellos.

Reglas de ámbito y orden de ejecución:

- Los componentes de tipo Sampler y Logic Controler son de tipo Orden (O), por lo que se ejecutan en la secuencia en que aparecen en el testplan (de arriba a abajo).
- Algunos componentes Logic Controler pueden cambiar el orden en que se ejecutan sus componentes anidados.
- El resto de elementos son de tipo Jerarquico (H), lo que significa que aplican solo a (TODOS) los componentes Samplers descendientes (hermanos, hijos, nietos, ...) de su padre
- Cuando varios componentes Jerarquicos (H) aplican a un sampler, hay un orden de ejecución que establece en que orden se le aplican:
  1. Configuration elements
  2. Pre-Processors
  3. Timers
  4. Sampler
  5. Post-Processors
  6. Assertions
  7. Listeners
- Si a un Sampler le aplican dos elementos con el mismo orden (según la regla anterior, por ejemplo dos Listeners), se le aplican en el orden en que aparecen en el testplan: de arriba a abajo
- Los elementos jerárquicos que se asocian a los samplers solo se ejecutan si se ejecuta el sampler. Si por ejemplo el sampler esta en un logic controler que hace que no se ejecute, no se ejecutan los elementos asociados.

Veamos el siguiente ejemplo poara ilustrar los conceptos anteriores:



Este ejemplo es un testplan con un sólo Thread Group. Hay cinco samplers (de nombres One, Two, Three, Four y Five). Hay dos logic controlers, de tipo Simple Controler, con este mismo nombre. Los controlers de este tipo solo agrupan, no alteran el orden de ejecución, por lo que el orden en que se ejecutan los samplers es de arriba a abajo: One, Two, Three, Four y Five.

Hay tres elementos jerárquicos: Timer#1, Assertion#1 y Timer#2. Además, por el tipo de estos elementos, se ejecutan despues de los samplers a los que afectan.

El elemento Timer#2 cuelga directamente del Thread Group, por lo que afecta a todos los samplers que cuelgan (directa o indirectamente) de este elemento, es decir, a todos los samplers.

El elemento Timer#1 cuelga del controller Simple Controller, por lo que afecta a todos los samplers que cuelgan de este mismo controllers directa o indirectamente, esto es, a Two, Three y Four.

El elemento Assertion#1 cuelga del sampler Three, por lo que sólo afecta a este.

Con lo anterior, el orden en que se ejecuta el testplan es (recordemos que si dos componentes jerárquicos del mismo tipo afectan a un mismo sampler, se aplican en el orden en que aparecen en el testplan de arriba a bajo):

```
One
```

```
Time#2
```

```
Two
Timer#1
Time#2
```

```
Three
Assertion#1
Timer#1
Time#2
```

```
Four
Timer#1
Time#2
```

```
Five
Time#2
```

El apartado 4.9 Execution order y 4.10 Scoping Rules del manual de usuario de JMeter explican esto mismo, pero con menos detalle.

Samplers para peticiones HTTP

Dedicamos una especial atención a los samplers que se utilizan para realizar peticiones HTTP a una aplicación, por ser el tipo de sampler que más se utiliza en JMeter, con diferencia.

Según se documenta en el apartado 18.1.2 HTTP Request, existen tres tipos de samplers para realizar peticiones HTTP:

- HTTP Request
- HTTP Request HTTPClient
- AJP/1.3 Sampler

El tercero sirve para probar un servidor Tomcat vía peticiones AJP (sobre HTTP). Este tipo lo ignoraremos en la discusión que sigue. Los otros dos son que se utilizan para que JMeter haga a una aplicación el mismo tipo de petición HTTP que haría un navegador.

Pero existen importantes diferencias entre los dos primeros a nivel de implementación de códígo fuente (por el contrario, a nivel de interfaz GUI el panel de control de ambos componentes es prácticamente igual):

El tipo de componente *HTTP Request* está basado en la implementación de HTTP que por defecto realiza Java (el JDK). A veces la documentación de JMeter se refiere a él como *default (Java) implementation*. Por el contrario, el componente *HTTP Request HTTPClient* está basado en la implementación de HTTP que realiza el framework Apache Commons HttpClient (http://hc.apache.org/httpclient-3.x/). Esta diferencia en la implementación de ambos provoca diferencias cruciales. Resumiendo, **la implementación Java por defecto del tipo de sampler HTTP tiene una serie de carencias / defectos que lo hacen inadecuado para su uso en pruebas de rendimiento**. Esto lo documenta el manual de usuario en 18.1.2 HTTP Request.

Grabación de un testplan (para aplicaciones web)

Existen básicamente dos estrategias para diseñar un testplan de JMeter (un fichero .jmx) para una aplicación web:

- A mano: creando cada componente mano en el lugar adecuado del árbol, y configurando el componente vía su panel de control.
- De forma automatizada: utilizando una herramienta que capture las peticiones HTTP que realiza el navegador, a medida que uno navega por la aplicación, y genere con cada una de estas peticiones HTTP un componente sampler en el árbol del testplan.

La primera es la estrategia que podríamos llamar de hacerlo todo a mano. La segunda, si bien no nos genera el testplan completamente, si al menos los samplers HTTP que el testplan tiene que ejecutar contra la aplicación. Además, la segunda requerirá que una vez creados los samplers HTTP con la herramienta que sea, modifiquemos la configuración por defecto de estos y añadamos nuevos componentes (timers, listeners, ...) que no crea la herramienta. En cualquier caso, la segunda estrategia ahorra trabajo.

Existen básicamente dos herramientas que nos permiten generar de forma automática los samplers HTTP de un testplan:

- El componente HTTP Proxy Server de JMeter
- La utilidad Badboy© (http://www.badboy.com.au/)

Ambas herramientas generan un .jmx, que como decimos más arriba, normalmente no es directamente utilizable por JMeter y suele requerir ajustes adicionales.

HTTP Proxy Server

El *HTTP Proxy Server* es un tipo de componente de JMeter, igual que cualquier otro componente como el HTTP Request o el

SMTP Sampler (por citar algunos). La diferencia respecto a éstos es que el HTTP Proxy Server sólo se puede crear en el Workbench de la interfaz GUI, con lo que no se graba en el fichero .jmx cuando grabamos el testplan vía la interfaz GUI.

Nota:

Hay también otros componentes muy útiles que sólo se pueden crear en el Workbench, ver en el menú emergente del Workbench la entrada Non-Test Elements.

El panel de control de este componente tiene un botón Start que al ser pulsado arranca un servidor proxy embebido en JMeter. Como cualquier servidor proxy, escucha en un puerto de la máquina por el que recibe peticiones HTTP de un navegador cliente, y las envía a la URL que estas indican. Cuando la URL (aplicación web) responde, devuelve la respuesta al navegador que hizo la petición. Esto es exactamente lo que hace el HTTP Proxy Server de JMeter, pero a diferencia de otros proxys, no cachea las peticiones HTTP ni las respuestas a estas por parte de la aplicación, sino que el procesamiento que realiza con las peticiones consiste en generar para cada una de ellas un componente HTTP Request (o HTTP Request HTTPCLient, según se configure).

La conclusión es que a medida que el usuario navega por la aplicación con su navegador cliente, el HTTP Proxy Server traduce éstas en componentes HTTP Request del testplan. Para ello evidentemente se necesita configurar el navegador utilizado para que haga peticiones al HTTP Proxy Server, en lugar de hacerlas directamente a la aplicación.

El panel de control del HTTP Proxy Server permite establecer cosas como:

- El puerto de escucha del proxy
- El tipo del sampler que debe generarse para cada petición HTTP (HTTP Request ó HTTP Request HTTPClient, recordar que la primera NO es adecuada para pruebas de rendimiento)
- El componente del árbol en que se crearán los samplers
- Si se deben excluir de la grabación peticiones que obtienen imágenes, hojas de estilo, scripts JavaScript u otras URLs o tipos de contenidos
- ...

En el enlace Recording Tests del sitio oficial de JMeter se puede descargar un tutorial que explica como configurar el HTTP Proxy Server y el navagador para capturar un testplan.

En el apartado HTTP Proxy Server del manual de usuario puede encontrarse la documentación de referencia de este componente.

En versiones de JMeter anteriores a la 2.4, el HTTP Proxy Server no podía capturar peticiones HTTPS, por lo que cuando la aplicación a testear utilizaba este protocolo, se requería otra herramienta como Badboy© (que si puede capturar peticiones HTTPS) para generar los samplers HTTP. En la versión actual de JMeter (2.4) no existe esta limitación. Aún así, el tutorial mencionado más arriba está sin actualizar, por lo que sigue dando la referencia de Badboy© cuando el protocolo es HTTPS.

Badboy©

Badboy© es una herramienta de Badboy Software©

Es una herramienta de testing de aplicaciones web, que funciona con Windows© e Internet Explorer©. Su filosofía consiste en ser un cliente de automatización de Internet Explorer© (utiliza el API de éste como motor de navegación).

Básicamente lo que hace la herramienta es grabar como un script (en un fichero con extensión .bb) la secuencia de navegacion por la web, que el usuario realiza con la propia herramienta. Una vez generado el fichero .bb, la misma secuencia de navegacion se puede volver a reproducir ejecutando el script .bb con la propia herramienta. La herramienta permite además modificar el script grabado introduciendo comprobaciones, condiciones, ...

Badboy© utiliza su propio formato de fichero (los .bb internamemnte no tienen nada que ver con los .jmx), por lo que un script grabado con Badboy© no es ejecutable con JMeter. Sin embargo, Badboy© dispone de la funcionalidad de grabar en formato .jmx una secuencia de navegación capturada. Esto es lo que lo hizo popular entre los usuarios de JMeter.

Para grabar como .jmx un script de Badboy, basta seleccionar la opción File / Export to JMeter ... en la interfaz GUI de Badboy©.

En la web de Badboy© (http://www.badboy.com.au/) puede obtener una distribución de la herramienta y acceder a su documentación online.

--

IMPORTANTE:

La versión 2.1 de Badboy© no genera bien los scripts de JMeter para la versión 2.4 de éste, aunque si que los genera bien para la versión 2.3 de JMeter (el formato de los ficheros .jmx ha cambiado ligeramente de la 2.3 a la 2.4).

Esto está documentado en el foro de Badboy: http://www.badboysoftware.biz/forum/viewtopic.php?p=8934&sid=335f6865bf6...

Hay dos workarounds a este problema:

1. Generar el fichero .jmx con la versión 2.1 de Badboy, cargarlo en la interfaz GUI con la version 2.3 de JMeter y grabarlo (con lo que se graba en un formato que puede leer la 2.4 de JMeter), y usarlo con la 2.4 de JMeter
2. No utilizar la version 2.1 de Badboy (es la estable), sino la version de desarrollo 2.1.1 (beta), que ya resuelve este bug.

--

Nótese que tanto el HTTP Proxy Server como Badboy© se limitan a capturar peticiones y convertirlas en samplers, todo lo más,

crean un componente de configuración para cada petición, en el que configura los datos de las cabeceras HTTP. Una vez grabado el .jmx, para que el testplan funcione como queremos, se requieren ajustes manuales, por ejemplo:

- Crear un componente HTTP Cookie Manager con el que el testplan pueda gestionar las cookies que envía la aplicación.
- Crear uno o varios componentes HTTP Request Default en que se configure el protocolo por decto, el puerto y el nombre del servidor que utilizan los samplers HTTP. Y modificar estos para que usen la configuración común que aporta el componente HTTP Request Default.
- Crear un componente User Defined Variables, configurarlo y modificar otros componentes para que utilicen las variables creadas
- Añadir uno o varios listeners, por ejemplo del tipo View Results Tree mientras construimos el testplan para depurar a medida que diseñamos
- ...

Construir planes configurables y mantenibles (propiedades, variables y funciones)

Propiedades, variables y funciones son conceptos relacionados en JMeter, y se utilizan para hacer que el testplan sea más mantenible y configurable.

Las propiedades y las variables se explican en el manual de usuario de JMeter en el apartado 4.11 Properties and Variables, y las funciones en el apartado 19. Functions.

Para construir un testplan configurable es necesario el uso de propiedades variables y funciones. El contruir planes de test que sean configurables es una buena práctica recomendada (ver Buenas prácticas y recomendaciones de uso). Este apartado explica los conceptos sobre propiedades, variables y funciones, y suministra un ejemplo de un testplan construido inicialmente de forma que no era configurable, y su versión configurable (ver Ejemplos).

Tanto las propiedades como las variables son valores, con un nombre asignado, que se mantienen en memoria durante la ejecución de JMeter, pero ambas se diferencian en la forma en que se crean y en el ámbito.

Las propiedades para JMeter son lo mismo que entendemos por propiedades en cualquier aplicación Java. En el caso de JMeter, las propiedades se crean de dos formas:

1. Especificándolas en el fichero jmeter.properties y los referenciados por éste (user.propertiers, system.properties, ...).
2. Especificándolas en la línea de comando al llamar a JMeter, con el modificador -q, -J, -G, -D (referencia: manual de usuario de JMeter, apdo. 2.4.5 Overriding Properties Via The Command Line)

NOTA:

Durante la ejecución de JMeter en modo GUI, se pueden ver las propiedades definidas en el proceso de JMeter con el elemento (componente) del workbench Non-Test Elements/Property Display.

Las variables son también valores con un nombre asignado, que se crean de alguna de estas formas:

- Especificándolas en el control panel del test plan (User Defined Variables,o UDV en adelante)
- Especificándolas en el componente "User Defined Variables" (Config Element)
- Con el componente "User Parameters" (Pre-Processor).
- Con el componente "Configuracion del CSV Dataset" (Config Element)

Determinadas funciones pueden crear variables (veremos cómo más adelante al hablar de funciones)

Propiedades y variables se diferencian en el ámbito, esto se explica en el subapartado Ambito.

Referenciar propiedades y variables

Si se ha creado una variable en un test plan, podemos referirnos a su valor con la sintaxis:

```
${NOMBRE_VARIABLE}
```

Así por ejemplo, si una variable de nombre HOST vale 127.0.0.1, en todos aquellos lugares del test plan (por ejemplo en campos del panel de control de un componente), donde se utilice la expresión ${HOST}, ésta se sustituirá por la cadena 127.0.0.1 .

Esto se utiliza para hacer el testplan más mantenible y configurable.

Las propiedades NO se pueden referenciar con la misma sintaxis que las variables, es decir, si user.dir es una propiedad, la sintaxis ${user.dir} NO sustituye la expresión por el valor de la propiedad. Sin embargo, se pueden usar las funciones property y P para obtener el valor de una propiedad (hablaremos más adelante de estas funciones).

Ambito

Además de en la forma en que se hace referencia a su valor, las propiedades y las variables se diferencian en el ámbito:

- Las propiedades son globales a JMeter (se definen en la JVM), por tanto, una propiedad tiene el mismo valor para todos los threads.
- Las variables son locales a cada thread: si una variable se cambia (veremos como al hablar de las funciones), SOLO se cambia la copia de la variable para el thread en el que se ha hecho el cambio. El resto de threads siguen teniendo el mismo valor para la variable.

Nota: hemos dicho A CADA THREAD, NO A CADA GRUPO DE THREADS (Thread Group).

Por ejemplo, si la propiedad user.dir tiene el valor /home/ecastilla, TIENE ESE VALOR EN TODOS Y CADA UNO DE LOS THREADS que se creen durante la ejecución del testplan.

Si una variable HOST se define con un componente User Defined Variables de un Thread Group, de 2 threads, con el valor 127.0.0.1, y uno de los threads cambia su valor por 192.168.2.10, el valor de la variable sólo ha cambiado para ese thread, para el otro sigue valiendo 127.0.0.1.

Importante saber que las UDV especificadas en el control panel del test plan o en un componente User Defined Variables tienen el mismo comportamiento (nos referiremos a ellas como variables UDV): se crean AL PRINCIPIO DE LA EJECUCION DEL TESTPLAN CON EL VALOR QUE SE ESPECIFIQUE (esto tiene varias implicaciones como veremos a continuación). Posteriormente cada thread creado durante la ejecución del testplan recibe una copia de cada una de esas variables, inicializada cada una con el valor que se haya especificado (referencia: manual de usuario de JMeter apdos. 4.11 Properties and Variables, 18.9.1 Test Plan y 18.4.13 User Defined Variables).

Implicaciones de lo anterior:

- En el control panel del testplan no podemos referirnos (con la sintaxis ${NOMBRE_VARIABLE}), a variables UDV definidas en el mismo panel ni en un componente Used Defined Variables
- Si una variable UDV se especifica con diferentes valores en varios componentes User Defined Variables, o una variable UDV especificada en el control panel del testplan se redefine en uno o varios componentes User Defined Variables, entonces el valor con el que se inicializa la variable es el último asignado según el orden de aparicion en el plan
- Si se necesita definir una variable UDV, utilizando otra(s), como parte del nombre o del valor, crear un componente User Defined Variable, en el que una o varias de las variables hagan referencia a otras UDV, por ejemplo definidas en el control panel del testplan
- En la definición de una variable UDV no se puede hacer referencia a otra variable que se crea durante la ejecución del testplan, porque en el momento en que la variable UDV se crea, aquella no existe todavía
- Si en la definición de una UDV se utiliza una función que devuelve un valor diferente cada vez que es llamada, solo el valor que devuelve la primera llamada a la función se utilizará en la definición de la variable, porque la función solo se la llamará una vez al principio del testplan
- Un thread puede cambiar el valor de una variable UDV (que a partir de entonces tendrá un valor diferente al de los otros threads), pero como norma general, se recomienda usar para estas cosas variables creadas en el thread, y dejar las UDV para guardar valores que no cambian durante una misma ejecución del testplan. Para definir variables durante la ejecución de un testplan (no al principio como sucede con las UDV), usese el componente User Parameters o Configuracion del CSV Dataset
- Aunque no es obligatorio con el componente User Defined Variables, pero por simplicidad se aconseja que las UDV se coloquen solo al principio del Thread Group o en el panel de control del testplan
- Las propiedades que existen en una ejecución de JMeter se pueden ver con un elemento: Non-Test Elements / Property Display (específico del Workbench). Con este elemento se pueden ver todas las propiedades definidas en una ejecución de la JVM, tanto las del sistema como las específicas de JMeter

Funciones

En JMeter existe un conjunto de funciones predefinidas, que pueden ser llamadas en casi cualquier sitio del testplan. En todos los sitios del testplan donde se puede especificar una variable, se puede especificar una llamada a función, por ejemplo, en los campos del panel de control de un componente.

La sintaxis de llamada a una función es

```
${__NOMBRE_FUNCION(ARG1, ARG2, ...)}
```

Donde NOMBRE_FUNCION es el nombre de la funcion (nos referimos a un nombre de función sin escribir los dos guiones del prefijo, ya asumimos que forman parte del nombre).

La lista de argumentos y el orden en que estos se especifican depende cada funcion concreta. Véase en el manual de usuario de JMeter, apdo. 19. Functions la referencia de todas las funciones predefinidas de JMeter.

Ejemplos de llamadas a funciones:

```
${__threadNum}
${__Random(1, 10)}
${__time(EEE, d MMM yyyy)}
```

Los nombres de funciones empiezan siempre por __ (dos underlines), para distinguirlas de los nombres de variables.

Cosas importantes que hay que saber sobre las funciones:

- Si en la llamada a una función no hay que especificar argumentos, no es necesario especificar los paréntesis
- Si un argumento contiene un caracter "," (como en la funcion time anterior), hay que anteponerle un caracter de escape (\) para indicar a JMeter que NO lo interprete como un separador de argumentos
- Para JMeter las funciones y las variables UDV (las que se definen en el control panel del test plan o en el componente "User Defined Variables") son lo mismo. Esto es lógico si observamos que la sintaxis para referenciar a una variable y para llamar

a una funcion son la misma (nombre de la variable o funcion entre ${ y })

- Si una funcion inexistente o una variable no definida son referenciadas, JMeter no genera ningún error, y devuelve el propio texto de la referencia. Por ejemplo, si la variable o funcion NOMBRE no está definida, y en algún campo del testplan se especifica la referencia ${NOMBRE}, JMeter NO sustituye nada, o lo que es lo mismo, devuelve ${NOMBRE} (sin las comillas) en el lugar de la referencia
- Si se define una variable UDF con el mismo nombre que una funcion, incluidos los dos guines de prefijo, entonces la variable sobreescribe a la funcion, lo que significa que como la sistaxis de referenciacion es la misma para variables y funciones, cuando se utilice, se sustituirá por el valor de la variable

Funciones importantes (la referencia de todas en 19. Functions):

property, P

Obtener el valor de una propiedad. Véase "Guardar un valor de propiedad en una variable".

Nota:

no esta documentado el por qué esta distinción entre propiedades definidas en los ficheros .properties y propiedades definidas en la línea de comando.

setProperty

Asignar un valor a una propiedad durante la ejecución del testplan. Véase Paso de valores entre threads y grupos de threads.

log, logn

Escribe en el log el mensaje que se especifique. Para asegurar que el mensaje es mostrado en la pantalla durante la ejecución del testplan, especificar OUT o ERR como segundo argumento. La funcion logn devuelve la cadena vacía, lo que significa que se puede utilizar en cualquier lugar de un campo de información del testplan sin afectarlo. Esta característica se pueden utilizar para tracear variables.

V

Las referencias a variables no pueden ser anidadas. Por ejemplo, ${Var${N}} no funciona. Para poder hacer esto existe la función V: la funcion V recibe como argumento un nombre de variable y devuelve el valor de esa variable (hasta aqui igual que usar ${VAR}). La peculiaridad es que como parte del nombre de la variable se puede especificar una referencia a variable. Por ejemplo:

```
${__V(VAR${N})}
```

Si N vale 1 devolverá el valor de VAR1, sin N vale 2 devolverá el valor de VAR2, ...

eval, evalVa

Evalúa una expresión de string, permitiendo interpolar referencias a variables y funciones en la string. La diferencia entre ambas es que eval recibe como argumento una expresión de string (nombre de variable o referencia a variable con ${ y }), mientras que evalVar recibe como argumento sólo un nombre de variable.

split

Trocea una expresión de string según un delimitador que se le pase, creando una variable con un sufijo distinto para cada item de la expresión.

JMeter Function Helper Dialog

En la interfaz GUI, la opción Options / Function Helper Dialog muestra un cuadro de diálogo que permite ver los parámetros que recibe una función, y construir la llamada a ésta. Una vez construido el texto de la llama con ayuda del cuadro de diálogo, éste se puede copir y pegarlo donde se necesite en el plan de pruebas.

Guardar un valor de propiedad en una variable

Para esto existen las funciones property y P.

La diferencia entre ambas es que la función "P" no permite asignar la propiedad a una variable, solo devuelve el valor de aquella al referenciarla, mientras que la funcion "property" si permite la asignación a una variable en la misma llamada a función. Por ejemplo:

```
${__property(abcd,ABCD,atod)}
```

asigna el valor de la propiedad abcd a la variable ABCD, y si no está definida esa propiedad, le asigna el valor atod

A pesar de esto, la funcion P se puede usar para especificar en un campo el valor de una variable.

Paso de valores entre threads y grupos de threads

Se hace aprovechando la característica de JMeter de que las propiedades son globales a todos los threads (de todos los grupos): basta asignar a una propiedad el valor de la variable que se quiere compartir entre los threads. Para hacer esto existe la funcion setProperty.

Componentes relacionados con variables

- User Defined Variables (Config Element)
- User Parameters (Pre Processor)

- Configuracion del CSV data set (Config Element)
- JDBC Connection Configuration / JDBC Request
- Sampler / Debug Sampler

Parametrizar el test plan usando variables, propiedades y funciones

El fichero adjunto eCOBandejas.zip contiene un script de JMeter en su versión no parametrizado, y parametrizado. Ver fichero LEAME.txt.

Esto se explica en el manual de usuario de JMeter, apartados 4.12 Using Variables to parameterise tests y 18.4.5 HTTP Request Defaults

Características

Ver la referencia Introducción a JMeter / Características.

Versiones

Ver la referencia Introducción a JMeter / Versiones recoomendadas

Buenas prácticas y recomendaciones de uso

- Permitir que los scripts de JMeter (ficheros .jmx) sean configurables
  Datos como direcciones IP, nombres de host, puertos, número de usuarios, tiempo de ejecución del testplan o número de iteraciones deberían ser propiedades configurable que el usuario pueda estabablecer antes de la ejecución del testplan. De esta forma los scripts .jmx no son específicos de los entornos en los que se despliegan las aplicaciones que testean
- No usar listeners innecesarios en el testplan
  El tiempo extra que los listeners emplean en procesar cada petición a la que afecta el listener, se contabiliza como parte del tiempo de ejecución de la petición. Esto influye negativamente en scripts .jmx que testean el rendimiento
- Capturar el fichero de sample result, y el fichero de log de cada ejecución en ficheros específicos, que contengan la fecha-hora como parte del nombre
- En pruebas de rendimiento, capturar el fichero de sample result en formato CSV
  El formato JTL contiene más texto porque es XML, por lo que JMeter requiere más tiempo para generarlos. Esto impacta negativamente en la ejecución de tests de rendimiemto
- Documentar la secuencia de interacciones que realiza un script .jmx generando un documento externo, por ejemplo un fichero de texto. De otra forma es muy dificil que el script sea modificable por quien lo haya generado. Esto cobra más importancia cuanto más complejas sean las interacciones que el script realiza con la aplicación

Ejemplos

El fichero adjunto *eCOBandejas.zip* contiene un script de JMeter en su versión no parametrizado, y parametrizado. Ver fichero *LEAME.txt*.

Enlaces externos

- Web oficial de la herramienta
- Artículos
- Manual de usuario de JMeter
- SoapUI
- Badboy
- Firebug
- Plugin de Maven para JMeter
- Plugin de Hudson para JMeter
- Varios plugins para JMeter
