Material descargable

Performance

egg

Este ejemplo analiza un sampler que realiza una petición a YouTube. A continuación, te proporcionamos el código junto con un desglose detallado de cada una de sus líneas para facilitar su comprensión. Si deseas ejecutarlo en JMeter, puedes crear un nuevo proyecto Maven en tu editor, copiar la clase, y luego utilizarla en un plan de pruebas en JMeter con las configuraciones necesarias.

```java
package com.example;

import java.net.http.HttpClient;
import java.net.http.HttpRequest;
import java.net.http.HttpResponse;

import java.net.URI;
import org.apache.jmeter.config.Arguments;
import
org.apache.jmeter.protocol.java.sampler.AbstractJavaSamplerClient;
import org.apache.jmeter.protocol.java.sampler.JavaSamplerContext;
import org.apache.jmeter.samplers.SampleResult;

public class YoutubeSampler extends AbstractJavaSamplerClient {

    @Override
    public void setupTest(JavaSamplerContext context) {
        // Configuración inicial si es necesaria
    }

    @Override
    public SampleResult runTest(JavaSamplerContext context) {
        SampleResult result = new SampleResult();
        result.sampleStart(); // Comienza el muestreo

        try {
            // Crear cliente HTTP
```

```
            HttpClient client = HttpClient.newHttpClient();

            // URL de la página web de YouTube
            String url = "https://www.youtube.com";

            // Crear y enviar solicitud HTTP GET
            HttpRequest request = HttpRequest.newBuilder()
                .uri(new URI(url))
                .GET()
                .build();

            HttpResponse<String> response = client.send(request,
HttpResponse.BodyHandlers.ofString());

            //Configurar el resultado de la muestra
            result.setResponseData(response.body(), "UTF-8");

result.setResponseCode(Integer.toString(response.statusCode()));
            result.setSuccessful(response.statusCode() == 200);



            // Mostrar solo los primeros 200 caracteres de la respuesta
en el mensaje
            result.setResponseMessage(response.body().substring(0,
Math.min(response.body().length(), 200)) + "...");

        } catch (Exception e) {
            result.setSuccessful(false);
            result.setResponseCode("500");
            result.setResponseMessage("Error: " + e.getMessage());
        } finally {
            result.sampleEnd(); // Finaliza el muestreo
        }

        return result;
    }

    @Override
    public void teardownTest(JavaSamplerContext context) {
        // Limpieza si es necesaria
    }
```

```
    @Override
    public Arguments getDefaultParameters() {
        Arguments args = new Arguments();
        // Este sampler no requiere parámetros, pero se pueden agregar
si es necesario en el futuro
        return args;
    }


}
```

Visión General del Código

1) **Imports**: Se importan las clases necesarias para manejar la solicitud HTTP, configurar el sampler, y gestionar el resultado de la prueba.

2) **setupTest**(): Este método se ejecuta antes de que comience la prueba. Aunque no se utiliza en este ejemplo, puedes agregar cualquier configuración inicial aquí.

3) **runTest**(): Este es el método principal donde se realiza la prueba:
   - Cliente HTTP: Se crea una instancia de HttpClient que manejará la solicitud.
   - URL: Se define la URL para YouTube (https://www.youtube.com).
   - Solicitud HTTP: Se construye una solicitud HTTP GET y se envía al servidor de YouTube.
   - Resultado de la muestra: Se guarda el cuerpo de la respuesta, el código de respuesta, y se marca como exitosa si el código de respuesta es 200.
   - Manejo de errores: Si ocurre un error durante la solicitud, la prueba se marca como fallida y se guarda el mensaje de error.
   - Finalización del muestreo: Se registra el final del muestreo.

4) **teardownTest**(): Este método se ejecuta después de que se completa la prueba, y se puede usar para limpiar recursos si es necesario.

5) **getDefaultParameters**(): Este método define los parámetros predeterminados que pueden ser configurados en JMeter. En este caso, no se requieren parámetros adicionales para este sampler, pero se puede modificar para incluirlos en el futuro.

Análisis Detallado Línea por Línea del Código:

```
@Override
public SampleResult runTest(JavaSamplerContext context) {
    SampleResult result = new SampleResult();
    result.sampleStart(); // Comienza el muestreo
```

SampleResult es una clase en JMeter que almacena los resultados de una prueba individual. Este objeto contendrá datos como el tiempo de respuesta, el éxito o fracaso de la prueba, los datos de respuesta, y más. Cada vez que se ejecuta una prueba (o "muestra"), se crea un nuevo **SampleResult** para registrar los resultados de esa prueba específica.

La línea **result.sampleStart()** indica cuando comienza la prueba en JMeter.

```
try {
    // Crear cliente HTTP
    HttpClient client = HttpClient.newHttpClient();

    // URL de la página web de YouTube
    String url = "https://www.youtube.com";

    // Crear y enviar solicitud HTTP GET
    HttpRequest request = HttpRequest.newBuilder()
        .uri(new URI(url))
        .GET()
        .build();

    HttpResponse<String> response = client.send(request, HttpResponse.BodyHandlers.ofString());
```

HttpClient es una clase en Java que se utiliza para enviar solicitudes HTTP y recibir respuestas.
newHttpClient() crea un cliente HTTP estándar con la configuración predeterminada.
Este cliente se usará para enviar solicitudes a un servidor y recibir respuestas.

String url nos guarda la información de la página web a la que queremos acceder. Esta URL será la dirección del servidor al que se quiere acceder para obtener la respuesta.

HttpRequest.newBuilder() crea un nuevo constructor para una solicitud HTTP (HttpRequest).
.uri(new URI(url)) establece la URI de la solicitud, donde url es la dirección del servidor.
.GET() indica que la solicitud es de tipo GET, que es el método HTTP usado para solicitar datos de un servidor.

.build() finaliza la construcción de la solicitud y devuelve un objeto HttpRequest que representa la solicitud completa.

---

client.send(request, HttpResponse.BodyHandlers.ofString()) envía la solicitud request usando el cliente client que se creó anteriormente.
HttpResponse<String> representa la respuesta HTTP recibida del servidor, donde el tipo de respuesta es String.
HttpResponse.BodyHandlers.ofString() indica que el cuerpo de la respuesta se debe manejar como una cadena de texto (String).

```
//Configurar el resultado de la muestra
result.setResponseData(response.body(), encoding:"UTF-8");
result.setResponseCode(Integer.toString(response.statusCode()));
result.setSuccessful(response.statusCode() == 200);



// Mostrar solo los primeros 200 caracteres de la respuesta en el mensaje
result.setResponseMessage(response.body().substring(0, Math.min(response.body().length(), 200)) + "...");
```

response.body() obtiene el cuerpo de la respuesta HTTP (el contenido de la página web que se ha solicitado).
"UTF-8" especifica la codificación de caracteres utilizada para interpretar el contenido de la respuesta.
El cuerpo de la respuesta se almacena en el objeto **result**, que representa el resultado de la muestra en JMeter.

---

response.statusCode() obtiene el código de estado HTTP (por ejemplo, 200 para éxito, 404 para no encontrado).
Integer.toString() convierte el código de estado numérico en una cadena de texto.
Esta cadena se almacena en result como el código de respuesta para la muestra.

---

response.statusCode() == 200 verifica si el código de estado HTTP es 200, que normalmente indica éxito.
Si es 200, la muestra se considera exitosa (true); de lo contrario, se considera fallida (false).
Este valor se almacena en result para indicar el éxito o el fracaso de la muestra.

---

response.body().substring(0, Math.min(response.body().length(), 200)) toma los primeros 200 caracteres del cuerpo de la respuesta.
Math.min(response.body().length(), 200) asegura que si el cuerpo de la respuesta tiene menos de 200 caracteres, no se intente acceder más allá de lo disponible.
+ "..." añade puntos suspensivos para indicar que la respuesta es más larga que lo mostrado.

El mensaje resultante se almacena en result como el mensaje de respuesta de la muestra.

```
catch (Exception e) {
  result.setSuccessful(success:false);
  result.setResponseCode(code:"500");
  result.setResponseMessage("Error: " + e.getMessage());
finally {
  result.sampleEnd(); // Finaliza el muestreo
```

result.setSuccessful(false); false indica que la muestra no fue exitosa.
Esto es útil en el contexto de pruebas de rendimiento para identificar que la solicitud o la operación que se estaba ejecutando no se completó correctamente.

---

result.setResponseCode("500"); "500" es el código de estado HTTP para "Internal Server Error", que indica que hubo un problema del lado del servidor.
Aunque este código se usa generalmente para respuestas HTTP reales, aquí se utiliza simbólicamente para indicar que ocurrió un error grave durante la ejecución de la muestra.

---

result.setResponseMessage("Error: " + e.getMessage()); e.getMessage() obtiene el mensaje de la excepción que fue capturada. Este mensaje suele describir lo que causó el error.
"Error: " es un texto adicional que se añade al inicio para aclarar que lo que sigue es un mensaje de error.
El mensaje completo se guarda en result como el mensaje de respuesta para que pueda ser revisado más tarde. Esto facilita la identificación del problema específico que causó la falla.
Por último, **result.sampleEnd();** Indica el fin de la prueba.