

Funcionamiento de CSV y counter

Para los casos de uso en los que jMeter es solicitante de datos externos hemos usado, tanto un archivo externo (CSV) como la utilización de la funcionalidad counter de jMeter.

Comenzaremos en primer lugar con la metodología que hemos utilizado para el uso de CSV.

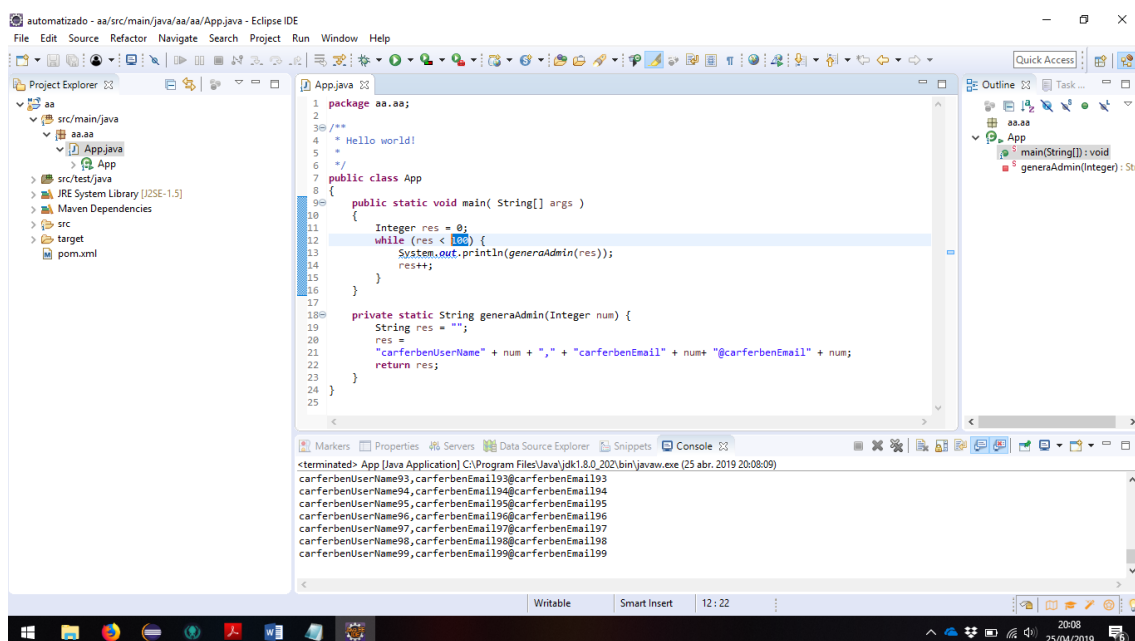
Para explicar el uso del archivo CSV nos apoyaremos en la prueba de rendimiento realizada con el programa de análisis jMeter en el que es registrado el usuario actor con la autoridad de hacker.

El problema por el que radica la necesidad del uso de archivos CSV incrustado en jMeter, así como de la variante del uso de un contador, como veremos posteriormente se crea a causa de que, cuando jMeter realiza una captura del tráfico generado por nosotros mismos en un primer uso de la aplicación, por ejemplo, en la creación de un hacker, existen parámetros que deben ser únicos, como por ejemplo el usuario y correo del actor. Estos datos se encuentran en una de las secuencias que captura jMeter, concretamente en un POST.

El problema, para concretar, es que en la ejecución de prueba de prueba de jMeter en su segundo loop intentará crear un nuevo usuario con el mismo usuario y correo que el anterior creado, y como debe ser único no podrá ser creado. Con la metodología del uso de CSV somos capaces de proporcionar a jMeter tanto un usuario como un correo inyectados a este a partir de un archivo externo.

¿Cómo realizamos este proceso?

En primer lugar, debemos crear el archivo CSV de forma externa a jMeter. Para ello hemos creado un pequeño método en Java en el cual tras indicarle un número de usuarios genera por consola una línea por cada usuario la cual cuenta con un nombre de usuario y un correo, separados por una coma. Cada usuario y correo es distinto ya que se usa una variable Integer añadida a la string de correo y a la string de usuario que aumenta en uno por cada línea.



```
1 package aa.aa;
2
3 /**
4  * Hello world!
5  */
6
7 public class App
8 {
9     public static void main( String[] args )
10    {
11        Integer res = 0;
12        while (res < 100) {
13            System.out.println(generaAdmin(res));
14            res++;
15        }
16    }
17
18    private static String generaAdmin(Integer num) {
19        String res = "";
20        res =
21            "carferbenUserName" + num + "," + "carferbenEmail" + num+ "@carferbenEmail" + num;
22        return res;
23    }
24 }
25
```

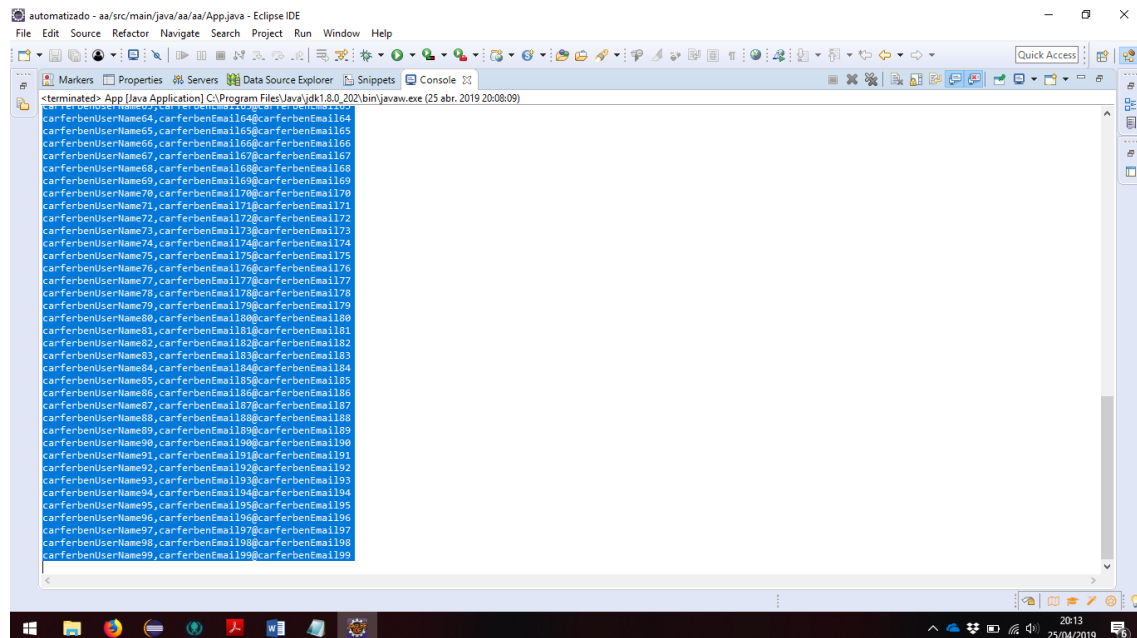
Console Output:

```
<terminated> App [Java Application] C:\Program Files\Java\jdk1.8.0_202\bin\javaw.exe (25 abr. 2019 20:08:09)
carferbenUserName93,carferbenEmail193@carferbenEmail193
carferbenUserName94,carferbenEmail194@carferbenEmail194
carferbenUserName95,carferbenEmail195@carferbenEmail195
carferbenUserName96,carferbenEmail196@carferbenEmail196
carferbenUserName97,carferbenEmail197@carferbenEmail197
carferbenUserName98,carferbenEmail198@carferbenEmail198
carferbenUserName99,carferbenEmail199@carferbenEmail199
```

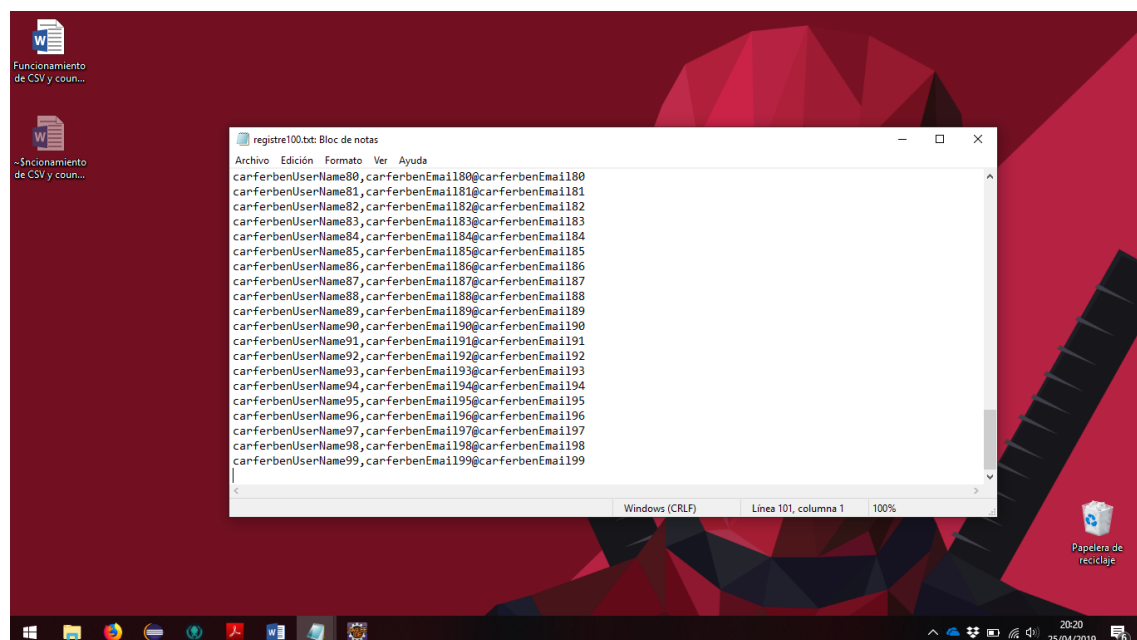
Como se observa en la imagen anterior, se ha realizado el pequeño método en java que nos generan las líneas solicitadas, el número anterior indica el número de líneas (y por tanto usuarios que son requeridos).

El pequeño método explicado se encuentra exportado en la ruta L03\Tests\hacker\register_hacker , el cual debe ser importado como proyecto Maven.

Una vez genera en la consola los datos que son necesarios copiaremos todos los valores de la consola y lo pegaremos en un bloc de notas, en un archivo que debe haber sido generado previamente con el nombre registre100.csv, siendo 100 variable en función del número de líneas que hayan sido generadas. (Ctrl A – Ctrl C – Ctrl V).



The screenshot shows the Eclipse IDE's console window. The title bar reads "automatizado - aa/src/main/java/aa/aa/App.java - Eclipse IDE". The console output displays a list of generated user data entries, each on a new line. Each entry consists of a username and an email address separated by a comma. The usernames range from "carferbenUser64" to "carferbenUser99", and the email addresses are in the format "carferbenEmail164@carferbenEmail164" to "carferbenEmail199@carferbenEmail199". The console window is titled "Console" and shows the output of the application.



The screenshot shows a Windows desktop with a Notepad window open. The Notepad window is titled "registre100.txt: Bloc de notas" and contains the same list of user data entries that were shown in the Eclipse console. The entries are copied and pasted into the Notepad window. The desktop background is a red and black geometric pattern. The taskbar at the bottom shows the Windows logo, several application icons, and the system clock displaying "20:20 25/04/2019".

Una vez insertadas las líneas en el archivo del bloc de notas guardaremos y cerraremos el programa.

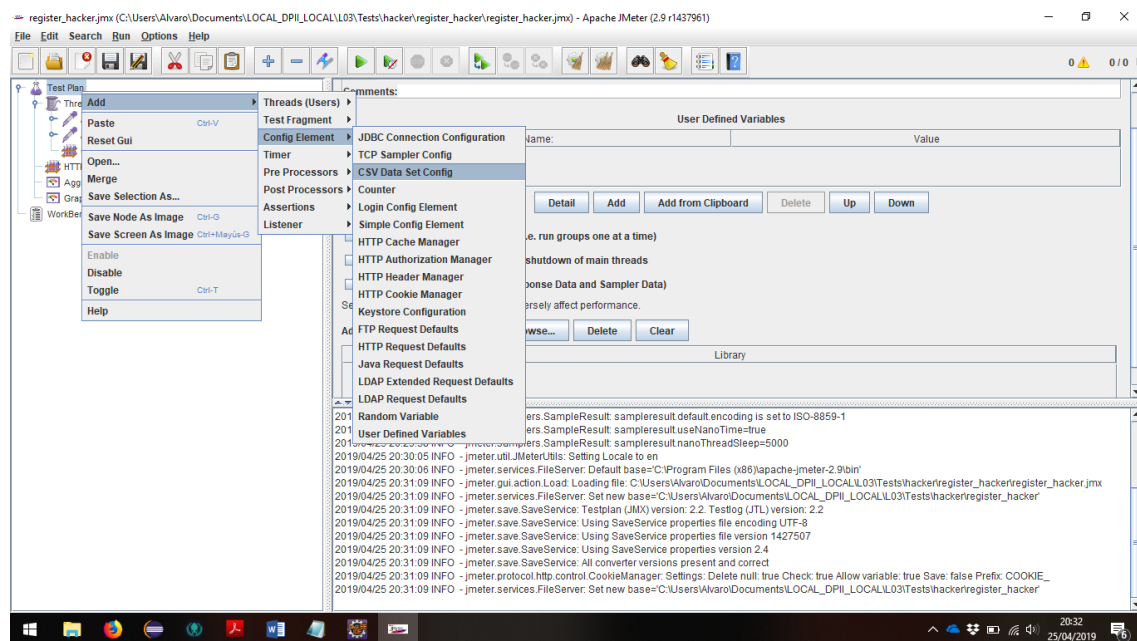
Tras cerrar este renombraremos nuestro archivo generado a registre100.csv, es decir simplemente cambiaremos la extensión del archivo renombrando este.

Este paso deberá ser realizado cada vez que cambiemos el número de usuarios, es decir, si aumentamos el número de hilos y/o el número de loops deberemos crear un nuevo archivo que tenga en cuenta esta nueva configuración. Como mínimo el archivo creado deberá tener un total de (numero de hilos) x Loops líneas, como mínimo.

A partir de este momento tendremos generado el archivo necesario para el uso de la metodología CSV.

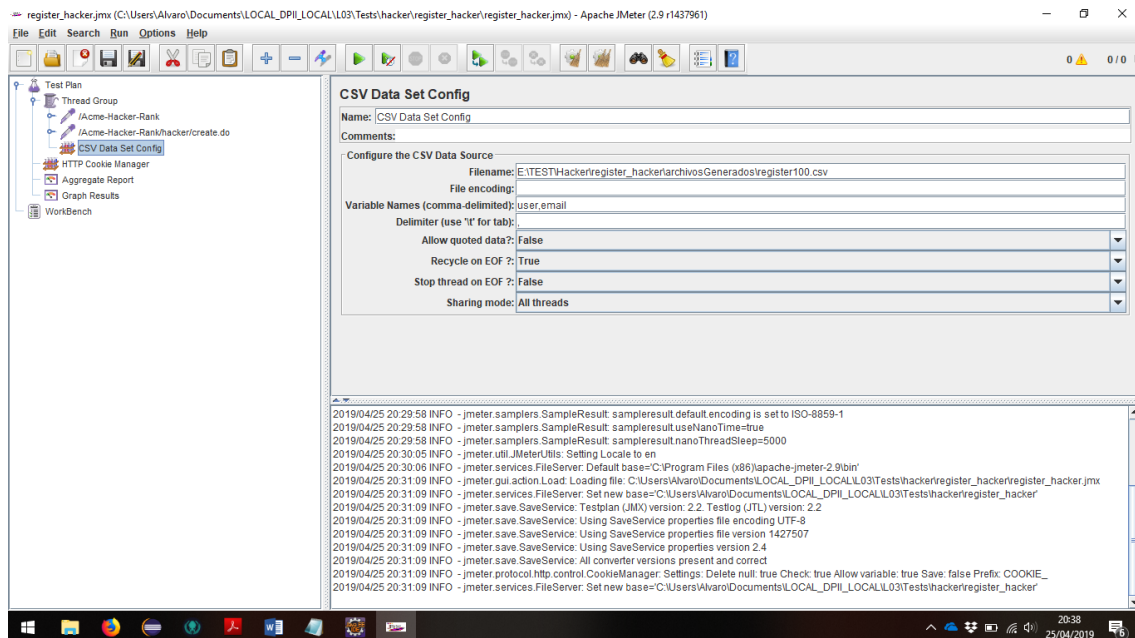
Pasamos al uso de CSV en el propio jMeter.

En primer lugar, usaremos un script de jMeter en el que ya se han capturado todas las secuencias del caso de uso del registro de un hacker. Una vez abierto deberá ser insertado el archivo CSV que hemos generado anteriormente.



La ruta necesaria para la inserción del archivo CSV es la mostrada en la imagen anterior. Sobre la configuración de hilos botón derecho, Add>Config Element>CSV Data Set Config.

Tras realizar esta acción aparecerá un archivo tal y como se muestra en la siguiente captura en el que deben ser configurados todos sus parámetros como aparecen en la misma.



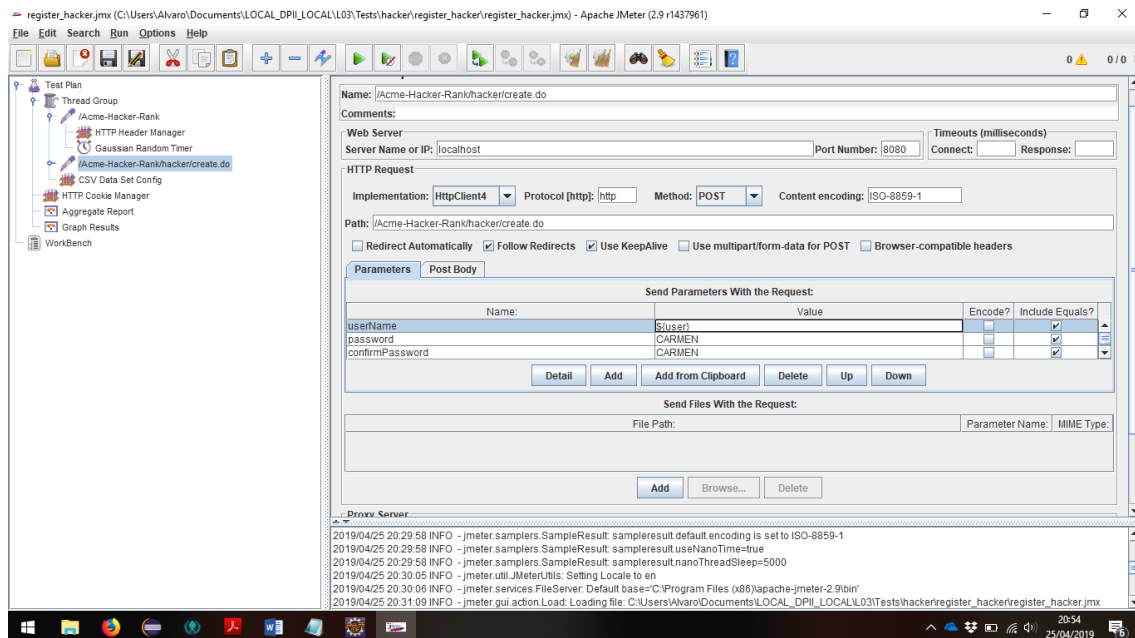
El primer parámetro que configurar es el parámetro Filename donde se le indica la ruta donde se encuentra el archivo .CSV

El segundo parámetro que configurar es Variable Names donde se le indica el valor que tiene cada variable de una línea en nuestro caso como tenemos dos variables debemos insertar dos nombres separados por una coma.

El tercer parámetro que configurar es el Delimiter que indica el carácter de separación para el nombre de las variables que han sido explicadas anteriormente.

El resto de los parámetros que se observan en la captura deben ser seteados de la misma forma que se observa en ella.

A partir de este momento ha sido completada la configuración del archivo .CSV, el siguiente paso es modificar las variables de la secuencia POST que deben ser únicas, para ello se debe realizar de la siguiente forma:



Debemos seleccionar la variable que queremos modificar y para acceder al archivo CSV debemos utilizar \${X} siendo X user o email en función de la variable que queremos insertar. Una vez realizada esta acción tendremos nuestro script de jMeter completamente configurado.

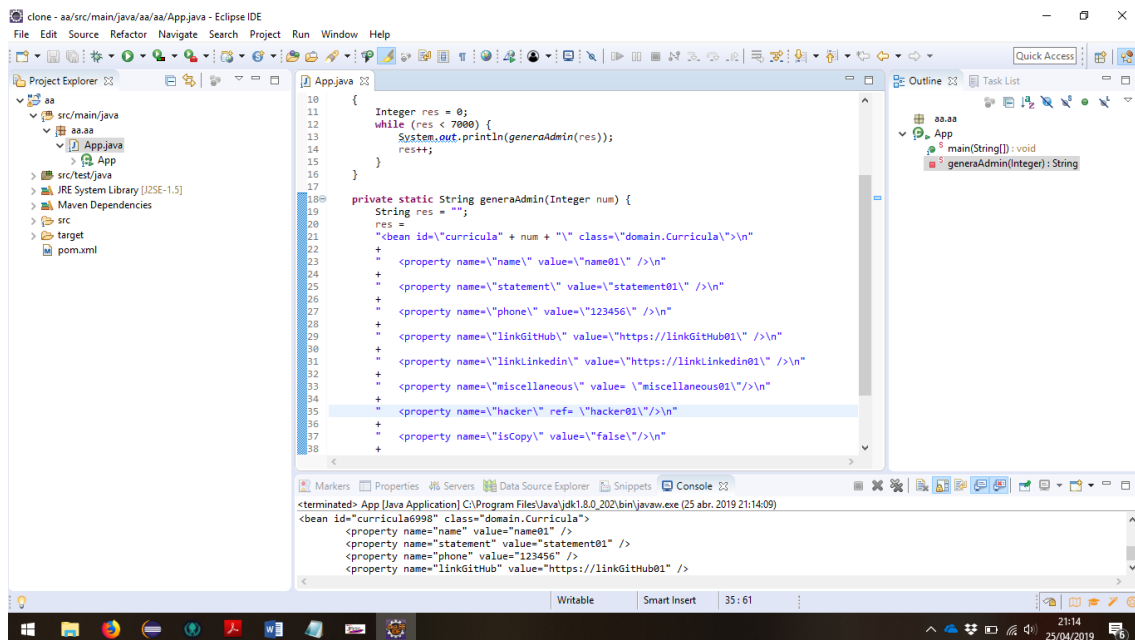
Tras esta configuración en la ejecución del script de jMeter cada vez que solicite la secuencia POST utilizará una de las líneas de del archivo CSV que le hemos proporcionado.

El segundo caso que queremos destacar es el uso de un Counter para la prueba de borrado.

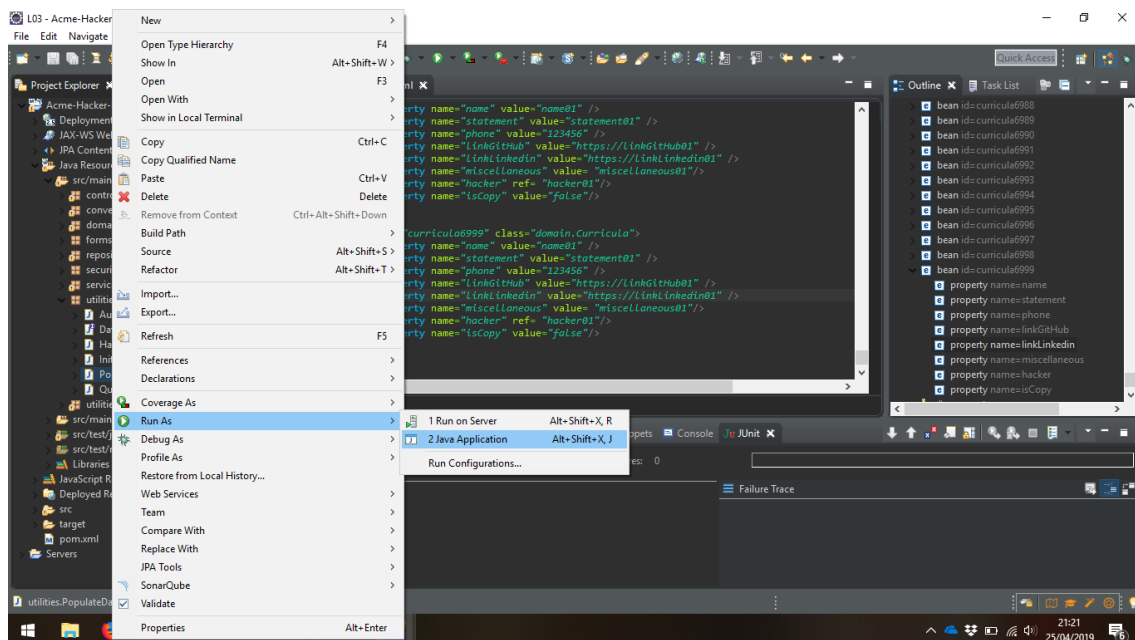
¿Cómo realizamos este proceso?

El primer paso es muy similar a lo descrito anteriormente, para la explicación de este proceso utilizaremos el ejemplo de borrar una curricula.

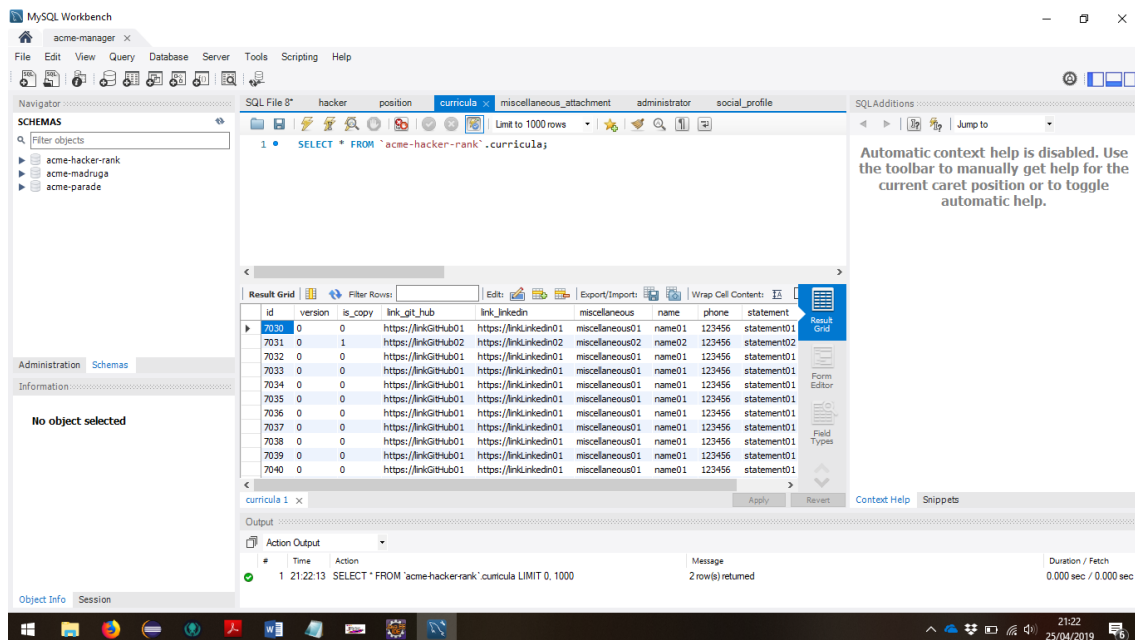
En primer lugar, deberemos crear un número de curriculas a borrar que será, concretamente, el número de usuarios multiplicado por el número de hilos.



Copiamos todas las entidades que han sido creadas en consola y la pegamos en nuestro populate.



Guardamos el archivo y ejecutamos poblando nuestra base de datos.

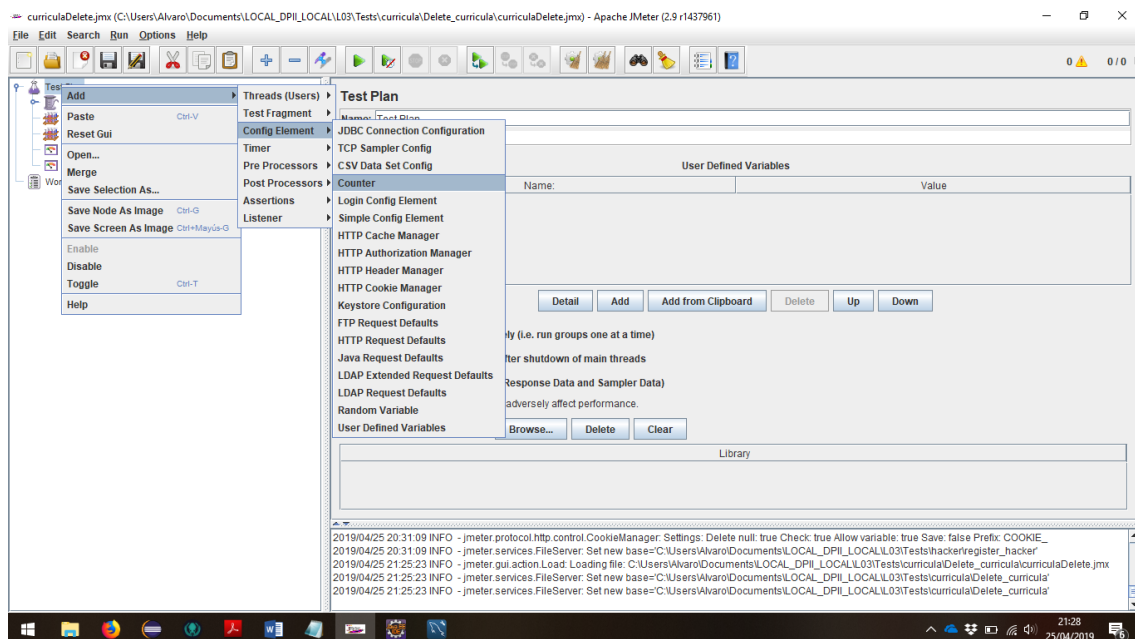


Una vez ejecutado nos dirigimos a nuestra base de datos y nos fijamos en la ID de nuestra primera entidad curricula.

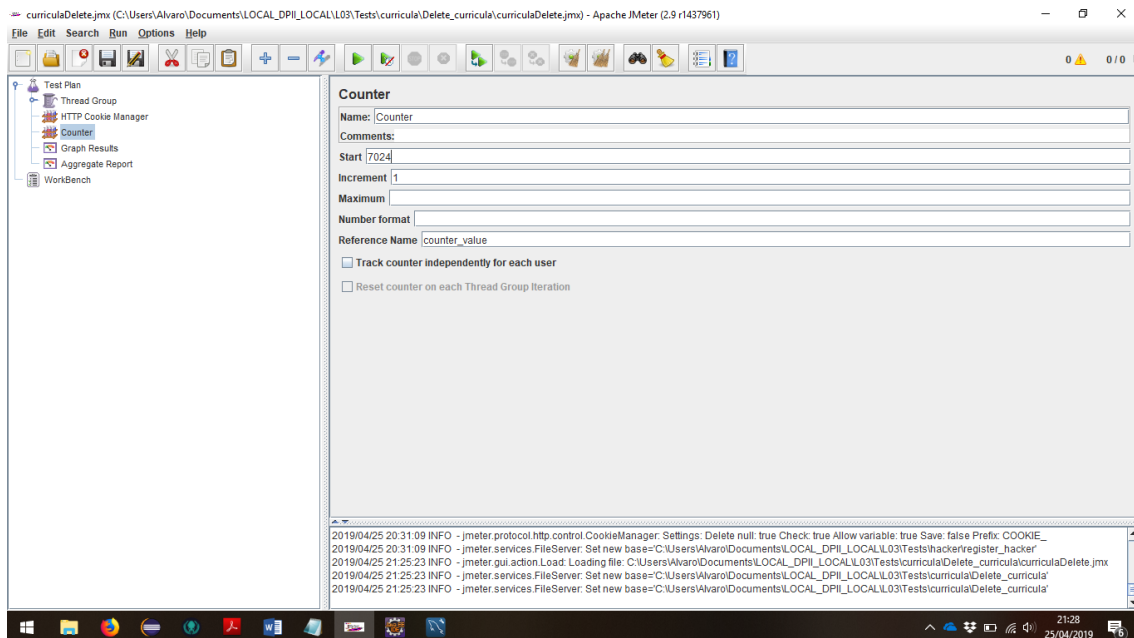
El paso siguiente es configurar nuestro script jMeter.

Utilizaremos un script en el que han sido capturadas todas las tramas necesarias para borrar una curricula, que en nuestro caso se realiza mediante un método GET.

El primer paso que debemos hacer es crear un elemento Counter.

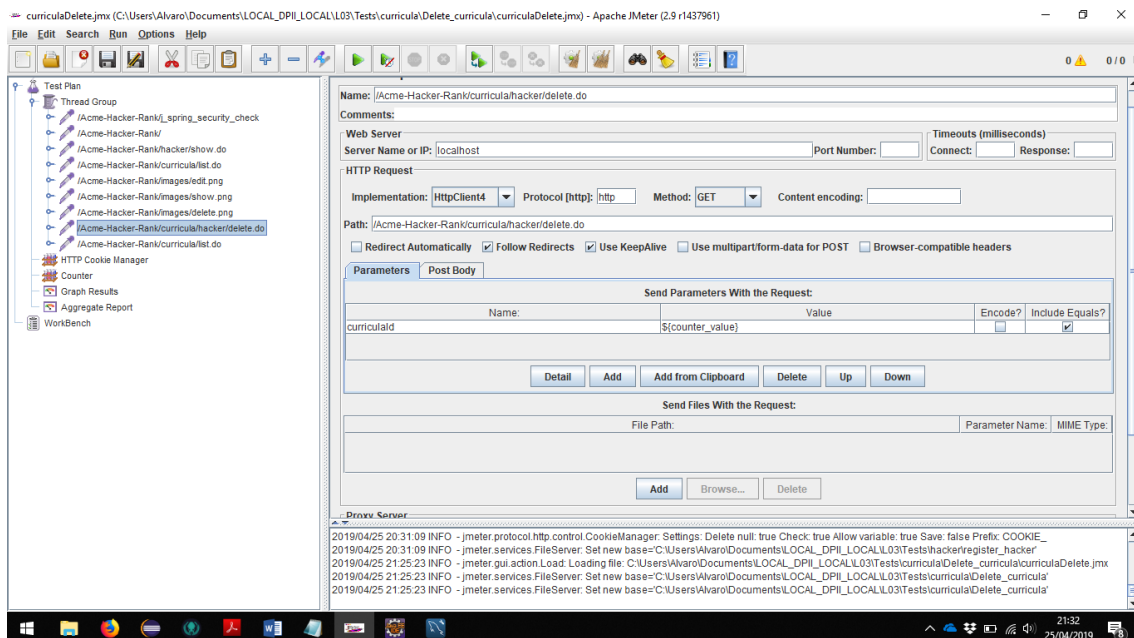


La ruta para la creación de un objeto counter es la mostrada en la captura anterior.



La configuración del elemento counter es la mostrada en la captura anterior, en start debemos poner la ID que indicamos anteriormente que debería ser observada en mi base de datos, en incremento lo setearemos como 1 y como nombre de referencia utilizaremos counter value.

Finalmente nos dirigimos a la captura del GET que realiza el borrado de currícula.



Modificamos el value del campo curriculaid y le insertamos la variable contador como \${counter_value}.

Una vez realizado esto por cada ejecución de un hilo jMeter solicitará una ID al elemento Counter que le cederá una ID válida.