Cómo construir un CGI con Clipper

Por Francisco García Aguado

Objetivo

Construir un CGI en Clipper, que siendo invocado desde un formulario de una página HTML, acceda a una base de datos en formato DBF, para realizar acciones de inserción de registros, mantenimiento y búsqueda.

¿Porqué Clipper?

El tema ha surgido para los que, como yo, no conocemos ni el PERL y muy poco el lenguaje C, ya que normalmente los CGI's suelen hacerse en dichos lenguajes con una *relativa* facilidad y efectividad. Así que, mientras no acometamos la tarea de aprender alguno de esos lenguajes de programación, con Clipper podemos dominar perfectamente el acceso a ficheros, índices, procesar la información etc. Con esto quiero decir, que esta solución que aquí expongo es puramente circunstancial y momentánea.

Ejemplo

Queremos que desde una página WEB un y a través de un formulario del tipo:

```
<form action="/CGI-DOS/PROGRAMA.EXE" METHOD=GET>
 <B>
 Su Email....:<INPUT TYPE="TEXT" NAME="email" MAXLENGTH="30" SIZE="30">
 <BR>
 Su Clave....:<INPUT TYPE="PASSWORD" NAME="clave" MAXLENGTH="9" SIZE="9">
 <BR>
 ¿Qué desea..?:
  <SELECT name="seccion">
   <option>  1 Direcciones SoftWare
   <option>  2 Direcciones Culturales
   .........
   .........
   <option> 30 Direcciones ONG
  </SELECT>
 <CENTER>
  <INPUT TYPE=submit value="Enviar">
  <input TYPE=reset  value="Borrar">
 </center>
<form>
```

Se devuelva a un usuario una página WEB en la cual se muestren las direcciones de Internet que tengamos en nuestra base de datos y que coincidan con su selección.

Primero debemos construir las bases de datos en formato DBF, con los campos adecuados. En el ejemplo que nos ocupa, las bases y los campos podrían ser:

USUARIO.DBF			
CLAVE	CARÁCTER	9	🔑
EMAIL	CARÁCTER	30	

SECCION.DBF			
CODIGO	CARÁCTER	3	🔑
NOMBRE	CARÁCTER	30	

URL.DBF			
CODIGO	CARÁCTER	3	🔑
URL	CARÁCTER	50	

El denominado PROGRAMA.EXE que aparece en el formulario, estará escrito en Clipper y su actuación esquemática, sería:

a) Recoge *el único* parámetro del *FORM* de la página WEB.

b) Formatea dicho parámetro que vendrá de la forma:

email=nombre@eidos.es&clave=76543&seccion=+30+Direcciones+ONG

Para formatear dicha cadena, utilizad las funciones que seguramente ya conoceréis: *STRTRAN()*, *STUFF()*, *SUBSTR()*, etc.

Ojo, pues si alguien introduce una Ñ/ñ o un acento en una vocal vais a recibir símbolos 'raros'. Un resumen breve de ellos es:

Símbolo	Carácter
%3A	:
%2F	/
%D1	Ñ
%F1	ñ
%E1	á
%E9	é
%ED	í
%F3	ó
%FA	ú

c) Una vez formateada dicha cadena/parámetro, tendré las variables locales de Clipper:

email
clave
seccion (sólo tres caracteres de la forma '30', por ejemplo)

d) Determina si el usuario existe o no, mediante su email y su clave. Si no existe, le da de alta en USUARIO.DBF (estamos suponiendo que estamos ante un servicio ABIERTO y que tan sólo pretendemos almacenar datos de personas que utilizan este servicio). Al dar de alta, deberemos crear un nuevo registro que deberá ser bloqueado momentáneamente (sobre bloqueos y demás ver libro *Programación en Clipper 5.X* de Grupo EIDOS, editorial RAMA, páginas 691 a 737 donde todo está muy bien explicado).

e) Busca en la base de datos *SECCION* el nombre al que corresponde los datos recibidos. Recordad que debéis indexar, para mayor rapidez.

f) Una vez determinada la sección (de la forma ya descrita, '30' por ejemplo), abrís las base de datos URL.DBF y vais buscando direcciones.

g) Aquí viene 'el truco', que si a alguien se le ocurre algo mejor, pues agradeceré que lo haga público. Esta parte del programa Clipper podría quedar más o menos así:

```
local linea:=8
archivo:="T"+alltrim(str(random(),7,0))+".HTM"
//random() es una función de la librería FIVEPRO de Antonio Linares
//Cañas, de libre distribución, que nos genera un número aleatorio
//Si lo deseaís, podreís crear vosotros un número pseudo-aleatorio
//utilizando la hora y fecha del sistema
use URL shared
set index to codigo
seek mcodigo          // declarar antes mcodigo
                      // con lo recibido del usuario EJ:mcodigo='30'
                      //podeís utilizar DBSEEK()
set device to print
set printer to &archivo

@ 0,0 say "CONTENT-TYPE: TEXT/HTML" //Estas líneas son imprescindibles
@ 1,0 SAY ""                        // para que la página WEB generada
@ 2,0 say "<HTML>"                  // sea leida por el Servidor.-
@ 3,0 say "<HEAD>"
@ 4,0 say "<title>Resultados de busqueda</title>"
@ 5,0 say "</HEAD>"
@ 6,0 SAY "<body.....................>"//Aquí poned lo habitual
@ 7,0 say "<b>Resultados de la Búsqueda....:</b><br>"

WHILE codigo=mcodigo

 @ linea,0 say URL+"<HR>"
 LINEA=LINEA+1
 SKIP

END
USE

@ LINEA,0 SAY "<HR><B>fin de direcciones</b>"
@ linea+1,0 say "</body></html>"
set device to screen
RUN MUESTRA.EXE &archivo // VER NOTA(*) SOBRE ESTA LLAMADA.-
delete file &archivo
quit
```

Bueno, ¿Qué hace este programa?. Realmente crea una página HTML lista para ser visualizada, grabándola momentáneamente en disco duro.

Si lo deseáis, podéis crear dicho archivo con las funciones *FCREATE()*, *FOPEN()*, etc... Cada uno que emplee la técnica que desee. La que aquí he expuesto es rápida, sencilla y funciona.

Dado que yo personalmente he tenido problemas con la salida estándar de Clipper (no resueltos, a pesar de múltiples consultas al respecto) lo que hago es lanzar la página generada hacia un sencillísimo programa escrito en C, con el cual os aseguro, no habrá problemas de

visualización. Dicho fuente podréis compilarle y enlazarle con cualquier compilador de C: Microsoft, Borland, etc., (veréis que la llamada la he hecho con un simple RUN).

Se puede utilizar una técnica mucho mejor y más compacta (que recomiendo) aprovechando que Clipper tiene un API de interfaz con C.

Para lo cual os invito a leer las páginas 857-912 del libro ya mencionado. En dichas páginas viene todo perfectamente explicado. De esta forma, creo un módulo objeto del programa en C que le enlazo con el de Clipper a la hora de crear el ejecutable, con lo cual sólo tendré un único EXE. Insisto, la explicación extensiva de esta segunda técnica, está perfectamente recogida en dicho libro.

El código en C del programa MUESTRA.EXE es:

```
#include <stdio.h>
#include <string.h>
#define NUM_CARACTERES_LINEA 90

main(int argc, char **argv)
{
  /**  variables **/
       FILE *temporal_f;
       char linea[NUM_CARACTERES_LINEA]="";
       char nombre_fichero[12]; /*8 caracteres del nombre, y 4 para extension*/
                                /* PUEDES METER MAS (EXTENSIONES HTML)
  /*cojo parametro*/
  strcpy(nombre_fichero,&(argv[1][0]));

  temporal_f=fopen(nombre_fichero,"r");

  while(fgets(linea,NUM_CARACTERES_LINEA,temporal_f)!=NULL)
  {
     printf(linea);
  }
  fclose (temporal_f);
}
```

Rendimiento global

Insisto, esto sólo está pensado para programadores, que como yo, no dominan C. Lo suyo, tal y como he hecho yo, es empezar a estudiar dicho lenguaje, que para programadores que ya tienen adquirida la lógica de programación en Clipper, os aseguro, no supondrá ningún problema. Sólo tiempo y ganas.

¿Es rápido este CGI?

El ejecutable Clipper ocupa mas o menos 170 Kb. En un servidor NT, la carga del programa se efectúa en mucho menos de medio segundo. El bucle de búsqueda de los datos, al estar indexados es muy rápido (calculad que un registro de una base de datos con 10.000 registros, indexada, se busca en apenas 0,5 segundos en un Pentium) El ejecutable en C es prácticamente instantáneo. Todo esto implica, que **es muy rápido dicho CGI** y estoy

convencido que serán las líneas telefónicas y su saturación, las que realmente ralentizarán el proceso, pero esto ocurre con C, con PERL o con cualquier otro lenguaje.

Si tenéis alguna duda, podéis poneros en contacto conmigo, vía EMAIL a la dirección paco@nemo.es, y en la medida de mis posibilidades os responderé a vuestras cuestiones.

Así mismo, si alguien que lea esto, ha conseguido con código *puro* en Clipper, hacer un CGI, que me lo diga, pues es una información para mí, muy valiosa (y supongo que para más personas).

Hago este comentario, ya que el hecho de que a mi personalmente no me hayan funcionado los CGI escritos en Clipper *puro*, sin la intervención de C lo he achacado a que la salida estándar de Clipper no es recogida adecuadamente por el servidor. Esto supone dos hipótesis que, insisto, nadie ha podido resolverme:

> ¿Es cuestión de la salida estándar de Clipper que no es válida para estos temas?
>
> ¿Es mala o insuficiente configuración del servidor, que impide que los ejecutables en Clipper generen páginas HTML visualizables?

Yo en estos momentos lo desconozco, de ahí que haya tenido que emplear la técnica aquí expuesta, para salir del punto muerto en el que estaba.