Curso de CGI

Autora: Lola Cárdenas Luque maddyparadox@wanadoo.es
Última actualización: 25 de febrero de 2001

Este curso pertenece a la web:
http://rinconprog.metropoli2000.com/

Curso de CGI

En este curso se explica qué es un CGI, cómo se hacen y qué cosas hay que tener en cuenta. Los temas a tratar no son demasiado largos, lo que hace de este curso más bien un mini-curso, del que el lector debe extraer las conclusiones pertinentes y ampliar lo aprendido a base de desarrollar programas propios.

Ultima modificación: 25 de febrero de 2001
Autor: Lola Cardenas Luque

Curso de CGI

1. Introducción

CGI son las siglas de *Common Gateway Interface*, o interfaz de pasarela común. Se trata de una especificación que va a realizar la función de interfaz o pasarela entre el servidor web y los programas, llamados programas CGI, haciendo uso del protocolo HTTP y el lenguaje HTML. Un programa CGI será aquel que cumpla la especificación CGI, es decir, interactuará con el servidor de acuerdo a unos principios establecidos en la especificación. Veamos cómo funciona esto.

Usualmente, cuando un navegador busca un URL, sucede lo siguiente. En primer lugar, el ordenador cliente contacta con el servidor HTTP. Este busca el fichero solicitado por el cliente y envía ese fichero. El cliente entonces visualiza el fichero en el formato apropiado.

Ahora bien, es posible instalar el servidor HTTP de forma que cuando un fichero de un directorio concreto es solicitado, ese fichero **no** sea devuelto. En lugar de eso, se ejecuta como un programa, y todo lo que el programa obtiene se envía de vuelta al cliente para ser visualizado. Obviamente, el directorio en el que están estos programas debe tener permiso de ejecución, así como los programas, y los permisos de lectura o de lectura/escritura para otros programas que pudieran usarse.

Resumiendo: los programas CGI son programas que se ejecutan en el servidor en respuesta a peticiones del cliente.

El servidor creará una información especial para el CGI cuando pasa a ejecutarlo, y esperará la respuesta del programa. Antes de que el CGI se ejecute, el servidor crea un entorno con el que trabajará el programa CGI. Este entorno comprende la traducción de cabeceras de peticiones HTTP en variables de entorno a las que podrá acceder nuestro programa. El resultado de la ejecución del programa suele ser una serie de encabezados de respuesta HTTP y HTML. Estos encabezados son recogidos por el servidor y enviados al cliente que hizo la petición.

Los programas o scripts CGI pueden escribirse en cualquier lenguaje de programación que sepa manejar entrada y salida estándar. La elección depende de qué nos gusta más, y un poco de sobre qué sistema operativo está el servidor. Si el servidor corre bajo una máquina Unix, a buen seguro podremos programar en C o en Perl (por ejemplo), solicitando al administrador del sistema (si no lo es uno mismo) que le de los permisos necesarios para poder ejecutar los programas. Si el servidor corre bajo una máquina Windows, también podremos programar en C o en Perl, esto último si el servidor tiene el intérprete instalado. Lo que programemos en C tendremos que compilarlo y poner el ejecutable en el directorio destinado a los CGI. Si usamos Perl o algún otro lenguaje interpretado, no tendremos necesidad de esto; simplemente pondremos nuestro script en el directorio para los CGI, y cuando se llame al CGI, el servidor se encargará de ejecutar el intérprete. Lo más recomendable es usar un lenguaje lo más portable posible, como los dos citados, pues el cambio de sistema operativo afectaría mínimamente al programa hecho y no tendríamos que cambiar muchas cosas.

En la sección Programas podeis descargar un sencillo servidor para Windows, llamado Xitami, con el que podeis empezar a hacer vuestros pinitos. Además, es fácilmente configurable. Os recomiendo leer la documentación que trae, porque cuenta muchas cosas interesantes.

Supongamos que sabemos hacer un programa CGI (cosa que empezaremos a aprender en el siguiente capítulo), que lo tenemos colocado en el sitio apropiado del servidor, que están claros los permisos para el programa... : ¿cómo lo llamamos desde una página HTML? Pues para ello tenemos varias posibilidades:

- Con la directiva <FORM>: si necesitamos que el CGI use la información que nos da un usuario al rellenar un form, hemos de indicarlo en el parámetro ACTION. Cuando pulsemos SUBMIT se llamará al programa CGI y se le enviará la información del formulario. Es conveniente que en ACTION especifiquemos una información válida.

??Con la etiqueta <A>. Por ejemplo, `<A HREF="/cgi-bin/fichero.cgi?vble1=valor&vble2=valor">Pulse por aquí</A>`

Esto significa que queremos que se ejecute el programa `'fichero.cgi'`, situado en el directorio `'/cgi-bin'` del servidor. Además, al pulsar sobre el enlace se le pasará la información de dos variables, cuyos nombres son `vble1` y `vble2`, que tomarán los valores respectivos `valor` y `valor`. El programa CGI recibirá la cadena `vble1=valor&vble2=valor`. Podemos ver que lo que hay tras el interrogante separa el nombre del programa CGI de los parámetros que va a recibir. Además, los parámetros y sus valores están separados por un signo igual, =, mientras que dos parámetros distintos vienen separados por un ampersand, &. Esto es importante de cara a cómo procesaremos la información que nos llega.

??Con la etiqueta <IMG>. Por ejemplo, `<IMG SRC="/cgi-bin/contador.cgi">`. Es un ejemplo típico de llamar a un programa que lleva la cuenta de las visitas, aumenta la cuenta y genera la imagen correspondiente, retornándola al cliente.


Ultima modificación: 25 de febrero de 2001
Autor: Lola Cardenas Luque

Curso de CGI

2. Las variables de entorno

En el capítulo anterior comentábamos que al llamarse a un programa CGI, el servidor creaba un entorno especial para dicho programa. Este entorno consiste en unas variables de entorno a las que asigna una serie de valores. Algunas de estas variables (las más usadas) son:

SERVER_SOFTWARE
Formato: nombre/versión
: Nos da el nombre y versión del programa servidor que atiende la petición de ejecutar el programa CGI.

SERVER_NAME
: Nos da el nombre de la máquina en la que se ejecuta el programa (el host), bien "traducido" (por ejemplo, www.unaMaquina.com), bien la dirección IP sin "traducir" (por ejemplo, 135.12.187.2, igual hasta he acertado O;-)).

GATEWAY_INTERFACE
Formato: CGI/revisión
: Nos da la revisión de la especificación CGI que cumple el servidor.

SERVER_PROTOCOL
Formato: protocolo/revisión
: Nos da el nombre y la revisión del protocolo de información con el que se ha creado la llamada al CGI

REQUEST_METHOD
: Nos dice con qué método se ha hecho la llamada. Por ejemplo, si lo llamamos desde un formularios, contendrá el valor GET o el valor POST.

QUERY_STRING
: Variable de entorno en el programa CGI recibe del cliente la cadena con los parámetros y sus valores. Por ejemplo, puede ser de este tipo: `vble1=valor1&vble2=&vble3=valor3` Recibiríamos esta cadena, y tendríamos que procesarla para usar esa información.

CONTENT_TYPE
: Se utiliza en llamadas al CGI que tienen información extra; por ejemplo, en el caso del método POST, esta variable contiene el tipo de los datos.

CONTENT_LENGTH
: Esta variable de entorno nos dice cuántos bytes ocupa la cadena recibida en la variable de entorno QUERY_STRING. Es necesario saberlo si vamos a reservar memoria para los datos de entrada.

Para más información sobre las variables de entorno, puedes consultar la siguiente dirección:
http://hoohoo.ncsa.uiuc.edu/cgi/env.html

Y en el próximo capítulo, nuestro primer CGI... sí, se trata de... ¡Hola Mundo!... ;-DDDD

Ultima modificación: 25 de febrero de 2001
Autor: Lola Cardenas Luque

Curso de CGI

3. El primer CGI: Hola Mundo!

Hemos de recordar que un programa CGI debe generar una salida que ha de ser reconocible por el cliente que la ha solicitado. Dado que los clientes suelen ser los navegadores, la salida del CGI será entonces un documento HTML, una imagen en formato GIF, un documento PDF, etc., y esto debe indicarlo de alguna manera: será enviando una primera línea en la que indique el contenido que va detrás.

Además, la salida que genera no será un fichero que se grabe en el servidor, sino que será una salida que irá a la salida estándar, es decir, el programa CGI escribirá "cosas" por la salida estándar y eso será lo que le llegue al cliente, quien tendrá que interpretarlo: si es un documento HTML lo visualizará, si es una imagen la cargará, si es un PDF es probable que nos solicite abrir el visor, etc.

Los ejemplos van a hacerse en lenguaje C; este lenguaje nos provee de la función `printf` para escribir sobre la salida estándar. Vamos a ver ya el ejemplo del `Hola Mundo!`:

```
/* Hola Mundo!
   Versión CGI - 20/2/2001 */

int main()  {
  printf("Content-Type:text/html\n\n");
  printf("<HTML><BODY><H2>Hola Mundo!</H2></BODY></HTML>");

  return 0;
  }
```

Lo compilamos, lo grabamos en nuestro directorio para los CGI, y lo llamamos desde el navegador. La forma de hacerlo depende del servidor. Por ejemplo, usando Xitami para Windows, no hace falta que renombres el fichero .EXE de niguna forma: para llamarlo desde el navegador escribes `http://ladireccionquesea/cgi-bin/HolaMundo`. Si usas el servidor Apache, tendrás que renombrar el fichero ejecutable a uno con extensión .cgi. Esto puedes hacerlo en el mismo momento de compilarlo, escribiendo: `gcc HolaMundo.c -o HolaMundo.cgi`. Lo colocas en el directorio para los CGI, y lo llamaríamos como sigue:

`http://ladireccionquesea/cgi-bin/HolaMundo.cgi`.

Como veis, es un programa en C normal y corriente. La primera línea es la que comentaba que es necesaria para que el navegador sepa qué información se le va a enviar. En realidad, son dos líneas, la primera, que debe llevar `Content-Type:tipo/subtipo` y la segunda en blanco, por eso tras escribir la primera línea está el `\n\n`.

En `Content-Type` podemos poner más cosas; vamos a ver una pequeña tablita con los tipos más frecuentes:

Encabezado	Significado
text/html	Este ya lo conocemos; se trata de un documento HTML
text/plain	Texto plano, es decir, ASCII puro y duro
image/gif	Imagen en formato GIF
image/jpeg	Imagen en formato JPEG, que puede tener una de estas extensiones: .jpeg, .jpg, .jpe
application/zip	Archivo comprimido en formato ZIP
audio/x-wav	Archivo de sonido en formato WAV
audio/midi	Archivo de sonido en formato MIDI, que tendrá la extensión .mid
video/mpeg	Archivo de vídeo en formato MPEG, que tendrá una de las siguientes extensioes: .mpeg, .mpg, .mpe
video/x-msvideo	Archivo de vídeo en formato AVI

Visto esto, podemos ir planteándonos crear unas funciones que podemos agrupar en una librería y que, por ahora, podrían ir escribiendo las cabeceras HTML en la salida estándar. Hecho esto, cuando fuéramos a escribir un programa CGI, quedaría como esto:

```
/* Esquema de CGI */

#include "cabeceras.h"

int main()  {
  /* Definición de variables */

  EscribeCabeceraHTML();

  /* Aquí está el trabajo que realiza el CGI */

  EscribePieHTML();

  return 0;
  }
```

Y en el fichero 'cabeceras.h' tendríamos las funciones 'EscribeCabeceraHTML' y 'EscribePieHTML', que se encargarían de toda la parte de sacar el 'Content-Type', <HTML>, <HEAD>, <BODY>, </BODY>, etc.

Para practicar con esto podeis ir creándoos vuestras librerías que saquen por salida estándar las cabeceras/pies. Además, dado que todo lo que va en un CGI es un programa absolutamente normal, podeis hacer, por ejemplo, que la salida genere una tabla por medio de bucles, u otras cosas que se os ocurran.

Nos vemos en el capítulo siguiente, en el que explicaremos cómo dar tratamiento a un formulario.


Ultima modificación: 25 de febrero de 2001
Autor: Lola Cardenas Luque

Curso de CGI

4. Tratamiento de formularios

Finalizamos este curso con una de las utilidades más grandes que tiene el uso de programas CGI: tratamiento de formularios.

Como ya sabemos, un formulario HTML está formado por un conjunto de directivas englobadas entre <FORM> ... </FORM>, que nos permiten poner cajas de texto, cajas de selección de opciones... y un botón que suele decir algo así como 'ENVIAR'. Cuando se pulsa este botón se ejecutará la acción que hubierámos especificado en el parámetro ACTION de FORM (para más información sobre creación de formularios, podeis echar una ojeada al capítulo 7 del curso de HTML de esta web.

ACTION puede tomar dos valores: un URL, o un 'mailto'. En el caso de 'mailto', los datos se envían directamente (sin procesar) a la dirección de correo electrónico especificada. El que nos interesa es el otro, es decir, un URL. Cuando especificamos un URL en ACTION, estamos diciendo la dirección (absoluta o relativa en la propia máquina) en la que está el programa CGI que se ejecutará.

Ahora bien, hay dos posibles maneras de enviar esta información, de las que hay que elegir una, y especificarla en el parámetro METHOD de la directiva FORM: GET y POST. La diferencia entre ambas posibilidades está en cómo se reciben los datos en el servidor.

Si elegimos el método GET, el servidor recogerá esta información en la variable de entorno QUERY_STRING. Al recogerla de esta forma, el tamaño de esta información es limitado, de hecho, no puede ser mayor de 1 Kbyte. Por el contrario, si elegimos el método POST, el servidor recoge los datos de la entrada estándar, lo que en principio no supone limitación alguna de tamaño, adecuado para recibir información de un TEXTAREA.

Vereis también que cuando se envía un formulario mediante GET, en la barra de título de navegador se verá algo como `direccion/programa.cgi?vble1=valor1&vble2=valor2`, es decir, NO se oculta la información enviada, durante un breve lapso de tiempo, que puede ser demasiado largo si enviamos, por ejemplo, una clave. Alguien más la podría ver. Sin embargo, cuando usamos POST nada de esto aparece en la barra de título: perfecto para claves y campos HIDDEN.

Es fundamental, por tanto, saber cómo averiguamos si el formulario ha sido enviado con GET o con POST: consultando previamente el valor de la variable de entorno REQUEST_METHOD. ¿Y cómo podemos hacerlo si estamos programando nuestro CGI en C? Con la siguiente función:

char *getenv(char *NombreVariableEntorno);

Le pasamos la cadena con el nombre de la variable de entorno, y nos devuelve un puntero a la posición del primer carácter de la cadena con los datos que tuviera dicha variable de entorno.

En el caso de que el método sea POST, tenemos una variable de entorno adicional que nos dice cuán largos son los datos que nos llegan por la entrada estándar, se trata de CONTENT_LENGTH. Si el método es GET, el tamaño de los datos de entrada será el tamaño de la cadena contenida en la variable de entorno QUERY_STRING, que era la que nos daba los datos en este método.

La siguiente función realiza la tarea de meter en una cadena la información del formulario, decidiendo qué hacer según el método con el que le hayan llegado los datos:

```
void Recibir_Parametros(char **parametros)  {
  int  longitud;
  char *QStr;

  if(strcmp(getenv("REQUEST_METHOD"),"POST")==0)  {
    longitud = atoi(getenv("CONTENT_LENGTH"));
    *parametros = (char*)malloc((longitud+1)*sizeof(char));
    gets(*parametros);
    }
  else  {
    QStr = getenv("QUERY_STRING");
    longitud = strlen(QStr);
    *parametros = (char*)malloc((longitud+1)*sizeof(char));
    strcpy(*parametros,QStr);
    }
  }
```

Usarla es sencillo: nos declaramos una variable de tipo char* en la función principal, y luego le pasamos su dirección a esta función.

Ahora que tenemos los datos, no hemos de olvidar que tenemos que tratarlos, puesto que los datos que recibimos tienen la forma:

```
vble1=valor1&vble2=valor2&...&vbleN=valorN
```

es decir, pares `variable=valor` separados por el símbolo &. Además, los espacios serán convertidos a un signo más (+), y los caracteres especiales (espacio incluido) vendrán codificados en la forma %HH, donde HH son dos dígitos hexadecimales. Será tarea nuestra decodificar estos datos para terminar obteniendo los valores correctos de las variables que recibamos en el CGI.

En la siguiente página:
http://www.ok.cl/cgi/downloads/cgiforms.htm podreis encontrar varias funciones para esto que os serán de utilidad.


Ultima modificación: 25 de febrero de 2001
Autor: Lola Cardenas Luque