Universidad Técnica Federico Santa María
Departamento de Electrónica
Valparaíso - Chile

“ Common Gateway Interface”

Nombre:	Mauricio Maluenda
Rol:	9821057-1
Profesor:	Agustín González
Fecha Entrega:	30-10 - 2002

INTRODUCCIÓN

A un nivel de abstracción por encima de los sockets ,nos encontramos con la combinación de HTTP y CGI. Esta tecnología es la que actualmente domina el desarrollo de aplicaciones distribuidas en internet. Desde la aparición de los browsers gráficos, el lenguaje HTML y el protocolo HTTP, los diseñadores disponen de una plataforma que estandariza al cliente (el browser) y permite generar vistosos resultados con gráficos en páginas HTML. Se estableció además un sistema de nombrado y localización de recursos y servicios : URL(Universal Resource Locator).Este sistema de nombrado es universal y estándar, lo que significa que no solo sirve para nombrar ficheros HTML, sino también servicios. Sin embargo, sin una interfaz que permita conectar esos nuevos servicios con un servidor WEB, el desarrollo de aplicaciones distribuidas no sería posible en este entorno.

Para ello se creó el interfaz CGI *(Common Gateway Interfac*e) que permite conectar a las aplicaciones servidor con el servidor WEB y les permite a las primeras generar paginas HTML, como respuesta a las peticiones de los clientes.

En este trabajo veremos temas como ¿Qué es CGI?,¿Qué son los scripts CGI?, la forma de comunicarse con estos scripts,¿Qué es el directorio cgi-bin?,sus variables de entorno, y todo lo relacionado con este tema...

Encontraremos además, sólo a modo de ejemplo, una interacción completa entre un cliente(a través de un formulario) y un servidor(en donde reside un script).El tema de la creación de formularios es tratada en el anexo de este trabajo, y la creación de scripts CGI, dado su extensión quese programan en distintos lenguajes, como C++,PERL,etc... no son tratadas en aquí y serán temas abordados por trabajos de otros compañeros.

¿Qué es CGI?

El CGI *(Common Gateway Interfac*e) es un estándar para comunicar aplicaciones externas con los servidores de información, tales como servidores HTTP o *We*b. Un documento en HTML que se trae del *Web* es estático, es decir, se mantiene constante: un fichero de texto que no cambia. Un programa CGI, por otro lado es ejecutado en tiempo real, así que puede generar información dinámica. Por ejemplo, supongamos que deseamos enganchar una base de datos de Unix al WWW, para permitir a gente de todo el mundo consultarla. Básicamente se necesitará un programa CGI que el *daemon* del *Web* ejecutará para transmitir la información al gestor de la base de datos, y recibir los resultados para presentárselos al cliente.

¿Qué son los scripts CGI?

Normalmente cuando un browser de Web (como por ejemplo el Netscape) llama a una URL en particular, sucede lo siguiente:

- Primero la computadora contacta el HTTP server con dicha URL (HTTP es el protocolo que se utiliza en las comunicaciones en la Web entre el server y el browser).
- El server HTTP revisa si el archivo requerido por nuestra computadora se encuentra en su sistema, en caso afirmativo envía el archivo como respuesta.
- Nuestra computadora entonces, muestra el archivo en el formato apropiado.

Además de todo esto, los servers de Web están configurados de tal manera que cada vez que se requiere un archivo de un directorio determinado (usualmente el "cgi-bin"), dicho archivo no es enviado; sino que es ejecutado como un programa y la salida de este programa es enviada a nuestra máquina para que ésta la muestre. Un ejemplo evidente puede ser el contador. El ordenador que entra en el sitio donde está el contador no hace nada sino visualizar la gif preparada por el servidor (siempre que sea un contador gráfico, que a través de un pequeño programa en C, "pegan" los distintos dígitos que representan la cantidad de accesos que ha tenido esa página en un único GIF), que lleva a cabo todo el trabajo de localizar el visitador para preparar una imagen con el número de los accesos. Esta función es conocida como "Common Gateway Interface" y los programas a los que nos referimos son llamados scripts CGI.

Un programa CGI se puede escribir en cualquier lenguaje que permita ser ejecutado en el sistema, como:

- C/C++,
- Fortran,

- PERL,
- TCL,
- algún Shell de Unix,
- Visual Basic,
- AppleScript ;

Simplemente depende de lo que tengamos en el sistema. Si usamos un lenguaje de programación como *C* o *Fortra*n, se debe compilar el programa antes de poder ejecutarlo.
Si miramos en el directorio */cgi-src*, encontraremos el código fuente de algunos programas CGI del directorio */cgi-bi*n. Pero, si usamos alguno de los lenguajes interpretados, como *PER*L, *TC*L, o un shell de *Uni*x, el script simplemente necesita residir en el directorio /cgi-bin, ya que no tiene un código fuente asociado.

¿Qué es el directorio cgi-bin?

Como dijimos, el directorio en el cual usualmente se encuentran los scripts CGI es el cgi-bin, de manera que los scripts que armemos deberán ser situados en el mismo. Si no tenemos control de la administración del server, debemos solicitar que se les dé permiso de ejecución a todos los archivos que pongamos en este directorio (cgi-bin).

El servidor conoce que este directorio contiene ejecutables que deberán ser ejecutados y su salida deberá ser enviada al navegador del cliente. No se puede simplemente crear un directorio *cgi-bi*n, el administrador del servidor deberá configurarlo para su uso. Si no está configurado, los scripst serán cargados como simples ficheros de texto.

Algunos servidores están configurados de tal manera que los ficheros con una determinada extensión (generalmente *.cgi*) son reconocidos como scripts y serán ejecutados como si estuvieran en un directorio *cgi-bi*n. La configuración de los directorios, o de la extensión mencionada antes, depende únicamente del servidor.

Como un programa CGI es un ejecutable, es equivalente a dejar a el mundo ejecutar un programa en nuestro sistema, que no es lo mas seguro a hacer. Por ello existen una serie de precauciones de seguridad que son necesarias de implementar cuando se usan programas CGI.

Probablemente lo que afectará al usuario típico del *We*b, es que hecho de que los programas CGI necesitan residir en un directorio especial(Por ejemplo, en la versión del servidor *HTTPd NCSA*, encontraremos un directorio denominado */cgi-bi*n.) Este directorio está generalmente

bajo el control del *webmaste*r, prohibiendo al usuario medio crear programas CGI. Hay otros métodos para permitir el accesos a scripts CGI, pero depende del *webmaster* que se de esta posibilidad. Así que debemos contactarlo para consultar la factibilidad de un acceso a los CGI.

¿Cómo puede un cliente desde su browser, especificar los parámetros que reconoce un programa CGI?

La respuesta está en los Forms. (Formularios, a pesar de que existen algunas aplicaciones que no necesitan de especificación de parámetros, como veremos más adelante)

A partir de la versión 2.0 de HTML, aparecida en 1995,se incluyó un nuevo "tag" en el lenguaje que permitía crear forms. Los formularios en HTML son muy parecidos a los formularios en papel, con campos para rellenar, elegir, etc....Dentro de una pagina HTML podemos crear un formulario, que será presentado al usuario por el browser, de tal manera que cuando éste presione el botón de "aceptar" (o submit en ingles) ,la información de los campos rellenados pasará al CGI que especifiquemos.

En una página, un formulario se encierra entre los tags <FORM> y </ FORM >. Éste tiene dos campos obligatorios:

- METHOD : puede ser GET o POST (métodos HTTP),que especifica como los datos del formulario entran en el programa CGI, bien mediante variables de entorno (en GET,la variable es QUERY STRING . Esta opción plantea problemas en muchos sistemas operativos (como MS_DOS y Windows 95/NT) ,que tienen una limitación (absurda por otra parte) en el tamaño de las variables de ambiente.), o bien a través de la entrada estándar del programa CGI (POST).
- ACTION: especifica el URL que recibirá los datos del formulario.

GET	POST
El CGI lo recibe por medio de una variable de entorno lo que significa una restricción en el tamaño de los datos.	El CGI lo recibe por medio de la entrada estándar (como si fuera teclado) lo que significa virtualmente recibir cualquier tamaño de datos.
El Browser envía los datos visiblemente haciendo una llamada de la forma: .../cgibin/cgi.pl?campo=algo Es la forma como se llaman cgis desde fuera de un formulario, por ejemplo los contadores de visitas son habitualmente llamados de la forma: <img src="Count.cgi?df=cgimaster2">	El Browser envia los datos directamente al CGI por lo cual no se ven en el browser (ideal para campos ocultos)
Al hacer un reload (recarga) de la página el browser pasará los parámetros automáticamente.	Al hacer un reload (recarga) de la página el browser pasará los parámetros previa confirmación

Supongamos que queremos enviar unas informaciones a nuestro CGI:

Nombre = Mauricio
Edad = 24
Ciudad = Valparaiso
Oficio = Estudiante de Electrónica

De forma que se busque en el database la persona (o las personas) que correspondan al criterio de nuestra búsqueda. Si pudiéramos indicar a QUERY_STRING sólo una clave, por ejemplo 'Marco', la búsqueda tendría éxito, pero se tendría que buscar entre muchos, quizás demasiados, resultados. La astucia de la que se hablaba antes es introducir, entre cada variable, el símbolo & y sustituir los espacios con el símbolo + de esta forma:

QUERY_STRING="Nombre=Mauricio&Edad=24&Ciudad=Valparaiso&Oficio=Estudiante+de+Electrónica"

En este trabajo veremos scripts que funcionan en base a los datos y variables que les son enviados a través de un FORM en una página HTML, y dejaremos para la siguiente parte de aquellos que no necesitan de esta entrada, por ejemplo los counters, las animaciones y otras cosas un poco más complicadas.

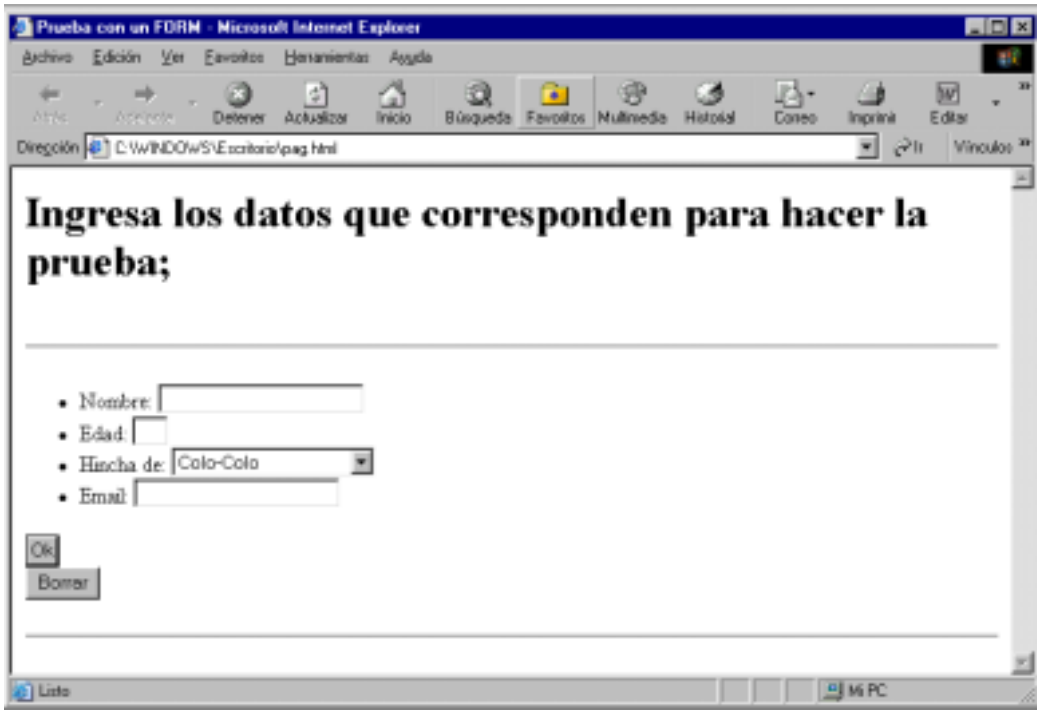
Un script que hace Eco de un Form.

Ahora vamos a solicitar algunos datos con un FORM apropiado, para posteriormente mostrar el script que permite retornar una página que contenga los nombres de las variables utilizadas junto con el contenido de cada una ingresado por el usuario.

Veamos el form que vamos a utilizar:

```
<HTML>
<HEAD>
<TITLE>Prueba con un FORM</TITLE>
</HEAD>
<BODY>
<H1>Ingresa los datos que corresponden para hacer la prueba;</H1>
<BR><HR>
<FORM METHOD="POST" ACTION="http://servidor.cl/cgi-bin/prueba.cgi">
<UL>
<LI>Nombre: <INPUT TYPE="text" NAME=nombre SIZE=20>
<LI>Edad: <INPUT TYPE="text" NAME=edad SIZE=2>
<LI>Hincha de:
<SELECT NAME="equipo">
<OPTION SELECTED> Colo-Colo
<OPTION> Colo-Colo
<OPTION> Universidad de Chile
<OPTION> Universidad Católica
<OPTION> Cobreloa.
<SELECT>
<LI>Email: <INPUT TYPE="text" NAME=email SIZE=20>
</UL>
<INPUT TYPE="submit" VALUE="Ok"><BR>
<INPUT TYPE="reset" VALUE="Borrar"><BR>
</FORM>
<HR>
</BODY>
</HTML>
```

La figura siguiente es lo que se observa en el browser:



Como ya mencioné, el script para hacer el eco será visto mas adelante.....

Una Interacción CGI Típica:

1. El usuario pulsa el boton “submit” del formulario. Esto hace que el browser recoja la información del form y la ensamble en cadena.
2. El browser invoca una petición del método al servidor en el URL especificado por ACTION. Esto hace que el browser cree una petición http y se la envíe al servidor.
3. El servidor inicializa las variables de entorno. Una vez que éste recibe la petición y se da cuenta del tipo de servicio que se pide , configura las variables de entorno que serán enviadas al programa CGI.
4. EL servidor http inicia el programa CGI. El programa a ejecutar es especificado en la petición del cliente.
5. El programa CGI recibe el cuerpo del mensaje, ya sea a través de la entrada estándar o a través de la variable de entorno QUERY_STRING.Tiene que identificar y separar cada campo del formulario.

6. El programa CGI hace su trabajo. Típicamente,utiliza recursos de bases de datos para dar respuesta a la petición. EL programa debe, entonces, dar su respuesta en HTML o en cualquier tipo MIME aceptable.(El script eventualmente retornar una página HTML o una imagen que es mostrada como resultado de la ejecución. Este último caso es el de los counters, Para diferenciar estos dos tipos de salida se envía un "header" al comienzo de la transmisión de la respuesta que las identifica según la siguiente especificación (extensiones MIME).

   Tipo de Información retornada Texto:

   Una página HTML / Content-type: text/html

   Una imagen .GIF / Content-type: image/gif, por ejemplo.
7. El programa CGI envía su respuesta a través de la salida estándar. Ësta es capturada por el servidor WEB y es enviada al cliente añadiéndole los headers pertinentes si el programa no lo hizo. Si lo hizo, simplemente pasa la respuesta al cliente(los programas que rellenan sus propios headers se llaman nph, non-parsed headers).
8. El servidor pasa la respuesta al browser del cliente.

Como se ve, el servidor inicia los programas CGI a petición de los clientes. Esto hace la interacción mucho mas dinámica . Sin embargo , nótese como el servidor tiene que lanzar un nuevo programa para cada petición.

Un script CGI debe:

- Ser ejecutable (o en el caso de Perl incluir la llamada al intérprete)
- Estar ubicado en el directorio cgi-bin. La ubicación de este server va a estar definida por el administrador del server.
- Sea capaz de acceder a los archivos que necesita. Como el script corre desde el directorio cgi-bin, todas las referencias relativas que hagamos van a estar referenciadas al mismo. Por las dudas, cuando tengamos que retornar una página HTML completa, usemos su URL

  completa (http://....) .
- Tener seteados los permisos correctamente. (Si no, nadie podrá ejecutar el programa excepto nosotros...)

Variables CGI:

Antes dijimos que el servidor http pasaba una serie de parámetros en variables de entorno y a través de la entrada estándar. También dijimos que los distintos métodos(POST y GET) indicaban la forma en que estos parámetros pasaban al programa CGI. Pues bien, los datos del formulario se envían de vuelta por el browser cuando el usuario pulsa uno de los botones de "submit".

En este sentido, los valores de cada campo están separados por el carácter "&" ,y se componen de pares <nombre,valor>,separados por "=" .Los espacios se sustituyen por + , y los + y otros caracteres especiales por "%xx" , donde "xx" son dígitos hexadecimales que contiene el valor ASCII del carácter. La aplicación de ejemplo mostrará como manejar este flujo de variables desde la entrada estándar utilizando PERL.

Variable	Valor
SERVER_ADMIN..	La dirección e-mail de la persona que se encarga del servidor
SERVER_SOFTWARE	Identifica el servidor y su versión.
SERVER_NAME	Identifica el nombre DNS del servidor, o ,en su defecto, su dirección IP
GATEWAY_INTERFACE	Versión del protocolo CGI utilizada, normalmente CGI/1.x.
AUTH_TYPE	Muestra el protocolo específico de autentificación.
CONTENT_LENGTH	La longitud del cuerpo del mensaje que envió el cliente. Es utilizado por el CGI para saber cuando dejar de leer de la entrada estándar.
CONTENT_TYPE	Tipo MIME de los datos del cliente.
HTTP_REFERER	El URL del que el script fue invocado.
REQUEST_METHOD	El metodo (get,post) utilizado por el cliente.
HTTP_USER_AGENT	User-Agent de la petición http.
QUERY_STRING	Los datos del form del usuario (si la petición es get).
REMOTE_ADDR	Dirección IP del cliente.
REMOTE_HOST	Nombre DNS del cliente.
SCRIPT_FILENAME	El valor del path completo al script CGI.
SCRIPT_NAME	Nombre del script.

SERVER_PORT	Indica el puerto al que la petición del cliente se lanzó.
SERVER_PROTOCOL	Indica el protocolo que el servidor está utilizando.

SCRIPTS:

A continuación veremos el script del form visto mas arriba.

El script que utilizaremos será muy simple y nos va a servir de base para futuros desarrollos.

Veamos paso a paso su estructura:

Primero llama al intérprete en caso de ser necesario (Perl).

Luego de incluir/cargar la librería, de alguna manera corre la función que separa QUERY_STRING en sus componentes individuales.

Asignamos dichos componentes a las variables que vamos a utilizar.

Imprimimos una salida, como en cualquier programa y.. Listo!

Aquí van los fuentes en cada uno de los lenguajes.

Script en Perl :

```
!#/usr/bin/perl
push(@INC,"/var/opt/ncsa/httpd/cgi-bin");
require("cgi-lib.pl"); <-- Incluimos la librería.
&ReadParse; <-- Separamos las variables.
$nombre = $in{'nombre'}; <-- Asignamos los valores que
$edad = $in{'edad'}; nos enviaron a las variables
$equipo = $in{'equipo'}; para utilizarlos.
$email = $in{'email'};
print &PrintHeader; <-- Imprimimos el Header.
print "<HTML>\n";
print "<HEAD>\n";
print "<TITLE>Aprendiendo CGI- Ejercicio</TITLE>\n";
print "</HEAD>\n";
print "<BODY>\n";
print "<HR>\n";
print "Tu nombre es ",$nombre,"<BR>\n";
print "Tenés ",$edad," años y eres simpatizante de ",$equipo,"<BR>\n";
```

```
print "Si alguien quisiera escribirte tu email es: ",$email,"<BR>\n");
print "<HR>\n";
print "</BODY></HTML>\n";
```

Script en C:

```
#include <stdio.h>
#include <errno.h>
#include <uncgi.h> <-- Incluimos la librería.
char *getenv();
main()
{
char *nombre; <-- Definimos las variables
char *edad; que vamos a usar.
char *equipo;
char *email;
uncgi(); <-- Separamos las variables.
nombre = getenv("WWW_nombre"); <-- Asignamos los
edad = getenv("WWW_edad"); valores que nos
equipo = getenv("WWW_equipo"); enviaron las
email = getenv("WWW_email"); variables.
printf ("Content-type: text/html\n\n"); <-- Header
printf ("<HTML>\n");
printf ("<HEAD>\n");
printf ("<TITLE> Aprendiendo CGI- Ejercicio</TITLE>\n");
printf ("</HEAD>\n");
printf ("<BODY>\n");
printf ("<HR>\n");
printf ("Tu nombre es %s <BR>\n",nombre);
printf ("Tenés %s años y eres simpatizante de %s <BR>\n",edad,equipo);
printf ("Si alguien quisiera escribirte tu email es: %s <BR>\n",email);
```

```
printf ("<HR>\n");
printf ("</BODY></HTML>\n");
}
```

Veremos ahora un tipo de scripts que no utilizan formularios. Como ejemplo veremos los contadores:

Counters.

Implementar un "counter" no es una tarea muy difícil. Básicamente se graba en un archivo el valor del contador, y cada vez que se hace un nuevo acceso a la página controlada se lee este archivo, se incrementa en uno el valor leído y se vuelve a grabar. Veremos un fragmento de script que hace esta tarea. Por otro lado no quería dejar de comentar que existen scripts más complicados que pueden generar una salida en la forma de un archivo .GIF. Estos scripts lo que hacen es leer el valor a representar, "pegan" en una misma imagen los dígitos que componen dicho número y la envían.

```
#!/usr/local/bin/perl
open(CONTADOR,"$counter.dat") || die "Error al abrir el archivo: $!\n";
$contador = <CONTADOR>;
close(CONTADOR);
if ($contador =~ /\n$/) {
chop($contador);
}
$contador++;
print "<HTML><HEAD><TITLE>Bla,bla,bla..</TITLE></HEAD>"
open(CONTADOR,">$counter.dat") || die "Error al cerrar el archivo: $!\n";
print CONTADOR "$contador";
close(CONTADOR);
```

¿Qué elementos hay que manejar?

Es difícil resumir los elementos a manejar en la programación CGI. Y esto es así porque el 99% de los servicios actuales en internet pasan por el WEB. En primer lugar, lo que hay que configurar es un servidor WEB. Este tipo de servidores , de cara a internet, están esperando en el puerto TCP 80 (hexadecimal), es decir, el puerto http, a que algún cliente conecte con ellos. Su

principal misión es retornar páginas HTML en servicio a las peticiones de usuarios. Por lo general, los servidores tienen un directorio que se llama “document root”(raíz de los documentos) que actúa como raiz de toda la estructura de directorios y páginas que se ofrecen a un posible cliente. Adicionalmente, se pueden configurar directorios pertenecientes a los usuarios del servidor (como en máquinas UNIX multiusuario). En UNIX, cada usuario tiene un directorio “home”, representado por la tilde(~)- Los servidores, por lo general, aceptan que se especifique que cada usuario tendrá un dirtectorio, llamado por ejemplo, “public_html” donde se localizarán todos los documentos propios del usuario.

Pero como hemos visto, la labor de un servidor WEB no es solo la de devolver documentos HTML para cada petición. Dentro de la raíz, se pueden configurar uno o varios directorios como “script path”(camino de scripts ejecutables). Estos directorios contienen programas ejecutables, es decir, programas CGI: el servidor debe seguir este protocolo para comunicarse con ellos. Por lo general, el directorio por defecto y mas comúnmente utilizado es clásico “/cgi-bin”. Hay realmente muchos problemas de seguridad asociados a este tipo de programas. Sobre todo cuando se considera la posibilidad de que los usuarios del host puedan incluir también programas ejecutables como parte de su interfaz con Internet.

A veces escuchamos la siguiente pregunta: ¿es más seguro escribir un CGI en Perl, en C o en otro lenguaje?'. Pues, no nos preocupemos porque la seguridad depende de *cómo* está escrito el CGI, no del *lenguaje* que se utiliza, aunque unos son más seguros que otros.

Hay que conseguir que los usuarios del sitio no interactúen NUNCA en primera persona con el intérprete Perl: si alguien pasase al intérprete perl unas líneas (después del '?') 'con mala intención', corremos el riesgo de encontrarnos con el disco vacío o de ver datos privados en las manos de otros.

Parece una tontería, pero es mejor hacerlo notar.

CONCLUSIONES:

La gran importancia de CGI radica en su capacidad de volver dinámicos elementos, páginas que anterior a su aparición solo eran estáticas, dado que ahora los archivos que son “pedidos” por el cliente no le son enviados en forma directa, sino mas bien, le son transferidos los resultados obtenidos al correr estos programas en el servidor. Además, posee una gran versatilidad, ya que permite transportar diversos tipos de archivos, como documentos de texto, documentos jpg, etc...que lo convierte en una poderosa herramienta en la creación de páginas interactivas. Es por esto que tiene un difundido uso en la actualidad , sus ventajas superan ampliamente a sus desventajas(como son las consideraciones de seguridad,por ejemplo) y su utilización es algo relativamente sencillo de aprender.

Vocabulario - Siglas.

URL: Universal Resource Locator Una nomenclatura que describe en forma compacta la ubicación de cada recurso en la Internet y el protocolo utilizado para acceder al mismo.

MIME: Multipurpose Internet Mail Extensions Una nomenclatura que permite identificar correctamente el tipo de datos que se esta enviando a través de una conexión.

HTML: HyperText Markup Languaje Un lenguaje para el "armado" de documentos de hypertexto incluyendo "links" y algunas otras características adicionales.

CGI: Common Gateway Interface Un mecanismo que permite a los browser de Web ejecutar programas en el server Web y recibir la "salida" de estos programas.

HTTP: HyperText Transport Protocol Un protocolo para la transferencia de documentos de hipertexto y otro tipo de archivos.

Bibliografía:

Tutorial de CGI ***http://www.cybercursos.net***

Tutorial “Programación pàra internet: PERL Y CGI”.- Diego Sevilla Ruiz

Tutorial de CGI www.abcom.com.ar

“ Common Gateway Interface ”

“Anexos”

ANEXO 1:"CREACIÓN DE FORMS"

Forms simples.

Comenzaremos con una página tipo que tiene incluido el tag correspondiente al inicio de un form;

```
<HTML>
<HEAD>
<TITLE> Página de Prueba </TITLE>
</HEAD>
<BODY>
<H1>Titulo Prueba de Forms </H1>

<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/prog.exe">
<-- aquí comenzaría el código para los elementos del form --!>
. . . . . .
. . . . . .
</FORM> <-- y aquí terminaría --!>

</BODY>
</HTML>
```

Evidentemente "prog.exe" es el nombre del ejecutable (script CGI) que correrá en el web-server una vez que el usuario envíe la información requerida, y POST es el método que se utilizará para realizar dicha acción. Existe otro método llamado GET que hace exactamente lo mismo, la diferencia radica en que este último envía la data dentro de la variable de entorno que habíamos mencionado (QUERY_STRING) "pegada" al URL del script y POST lo hace como un "stream" continuo de caracteres. Es preferible utilizar POST ya que el mismo no tiene limitaciones de tamaño y en cambio GET sí puede tenerlas en el server.

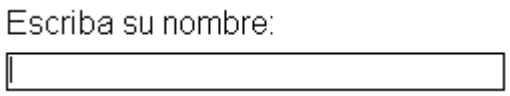
Text Line

El elemento más simple que podemos utilizar en un FORM es el "Text line" que le permite al usuario ingresar texto en una línea. La sintaxis utilizada para este tag es la siguiente;

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/text.exe">
Escriba su nombre<BR>
```

```
<INPUT TYPE="text" NAME="nom_var" SIZE=20 >
</FORM>
```

En el browser se verá:



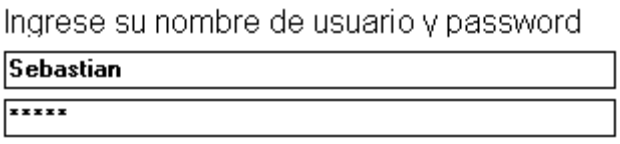
Cuando el usuario oprima la tecla "Enter" lo que haya escrito ser enviado al web-server. Veamos los parámetros de este tag. En NAME va el nombre de la variable que almacenará el texto que ingrese el usuario y en SIZE tenemos 20 que es la longitud en caracteres que va a tener la caja. Por medio de un script-CGI "capturaremos" la variable "nom_var" y leeremos su contenido, el cual podremos procesar posteriormente.

Password Text.

En adición a Text Line existe el tipo "password" que sirve para ingresar claves o textos secretos, cuya sintaxis es muy parecida. No se trata para nada de un medio seguro para enviar información a través de la Net, pero bueno, puede servirnos para este fin didáctico.

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/pass.exe">
Ingrese su nombre y password.<BR>
<INPUT TYPE="text" NAME="nom_var" SIZE=20 > <BR>
<INPUT TYPE="password" NAME="pass_var" SIZE=20 > <BR>
</FORM>
```

Obtendremos la siguiente salida :

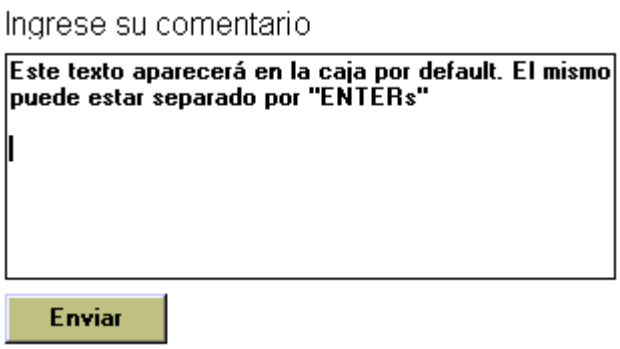


Obviamente lo que escribamos en la caja de password no aparecerá como texto normal sino con una "*" representando cada carácter.

Text Boxes

Si necesitamos recibir más de una línea podemos utilizar los tags <TEXTAREA ..> </TESTAREA> de la siguiente manera;

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/text.exe">
Ingrese su comentario<BR>
<TEXTAREA NAME="caja" ROWS=5 COLS=40> Este texto aparecerá en la caja por default. El mismo puede estar separado por "ENTERs".</TEXTAREA> <BR>
<INPUT TYPE="submit" NAME="boton_env" VALUE="Enviar">
</FORM>
```



El significado de cada parámetro es el siguiente:

- NAME Nombre de la variable que almacena el texto de la caja.
- ROWS Cantidad de filas de la caja de texto.
- COLS Cantidad de columnas de la columna.

Submit Button

En el caso que acabamos de ver, el "Enter" sirve para pasar a la línea siguiente mientras se está escribiendo, de manera que implementamos otro m,todo para enviar la información, que es el "Submit Button", cuando el mismo sea oprimido el texto que hayamos cargado será enviado.
Estos son los parámetros del "Submit Button":

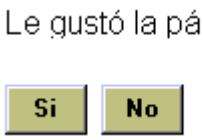
- NAME nombre del botón (para la referencia en el script)

- VALUE texto que aparece en el botón (por default es "Submit")

El parámetro "NAME" no es necesario en el caso anterior, pero es interesante citarlo ya que podemos llegar a utilizarlo en el siguiente situación;

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/text.exe">
Le gustó la página<BR>
<INPUT TYPE="submit" NAME="boton_env" VALUE="Si">
<INPUT TYPE="submit" NAME="boton_env" VALUE="No">
```

Se obtendría algo como:



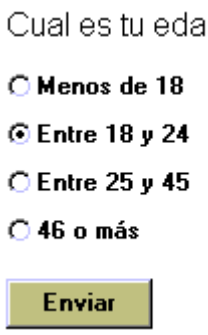
Aquí la variable boton_env contendrá los valores "Si" o "No" según cual de los dos botones oprima el usuario.

Radio Buttons

Los Radio Buttons permiten que el usuario elija una única opción entre varias. Son esos pequeños círculos que aparecen cuando tenemos que indicar el sexo, o bien un rango de edades o una opinión (muy bueno, bueno, malo). Veamos un ejemplo del código.

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/radio.exe">
Cual es tu edad?<BR>
<INPUT TYPE="radio" NAME="edad" VALUE="a"> Menos de 18<BR>
<INPUT TYPE="radio" NAME="edad" VALUE="b" CHECKED> Entre 18 y 24<BR>
<INPUT TYPE="radio" NAME="edad" VALUE="c"> Entre 25 y 45<BR>
<INPUT TYPE="radio" NAME="edad" VALUE="d"> 46 o más<BR>
<INPUT TYPE="submit" NAME="boton_env" VALUE="Enviar">
</FORM>
```

La pantalla mostraría algo como esto:



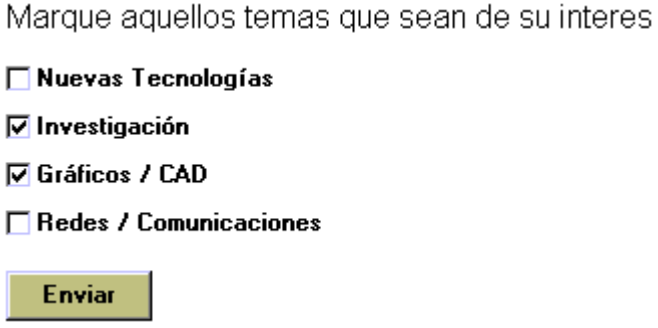
El form retornar dentro de la variable "edad" el valor a,b,c o d según corresponda la opción que haya sido elegida por el usuario. La cláusula CHECKED permite tener un ítem seleccionado por default.

Check Boxes

Los Check Boxes sirven cuando necesitamos recibir un input con más de una opción seleccionada, se utilizan por ejemplo para marcar las características que más nos agradan de un determinado producto. La sintaxis es bastante parecida al anterior.

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/check.exe">
Marque aquellos temas que sean de su interes:<BR>
<INPUT TYPE="checkbox" NAME="temas" VALUE="ntecs"> Nuevas Tecnologías<BR>
<INPUT TYPE="checkbox" NAME="temas" VALUE="inves" CHECKED> Investigación<BR>
<INPUT TYPE="checkbox" NAME="temas" VALUE="grafs" CHECKED> Grá ficos / CAD<BR>
<INPUT TYPE="checkbox" NAME="temas" VALUE="redes"> Redes / Comunicaciones<BR>
<INPUT TYPE="submit" NAME="boton_env" VALUE="Enviar">
</FORM>
```

En el browser tendríamos:



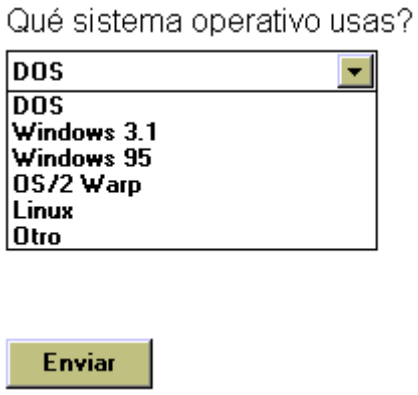
Igual que con los radio buttons, la cláusula CHECKED permite tener marcados algunos cuadraditos por default. En la variable "temas" van a ir a parar aquellos opciones que sean marcadas por el usuario.

POP UP list

Podemos también utilizar menús descolgantes como manera de elegir una opción entre varias mediante el tag <SELECT> como podemos ver más abajo.

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/popup.exe">
Que sistema operativo usas?<BR>
<SELECT NAME="sistema">
<OPTION SELECTED> DOS
<OPTION> Windows 3.1
<OPTION> Windows 95
<OPTION> OS/2 Warp
<OPTION> Linux
<OPTION> Otro
</SELECT> <BR>
<INPUT TYPE="submit" NAME="boton_env" VALUE="Enviar">
</FORM>
```

Si presionamos con el mouse sobre la flechita la lista desplegable se verá así:



Al igual que en los otros ejemplos, la variable "sistema" almacenará el ítem elegido.

Forms Multiples y el Reset Button

Todos los elementos que hemos mencionado pueden ser utilizados individualmente con un submit

button (o sea con un form para cada uno) o bien pueden ser empleados varios de ellos en una misma página.

Nota: Se estila en la creación de un buen form, el poner líneas horizontales arriba y abajo del form (<HR> o alguna línea gráfica) y proveer instrucciones de como se deben llenar los blancos.

Para finalizar construiremos un ejemplo que contenga todos los tags vistos hasta ahora. Este form múltiple introduce el uso de un nuevo tipo de botón, el Reset Button, que en resumida cuentas borra los datos que hayamos ingresado y deja los elementos en su opción de default.

```
<FORM METHOD="post" ACTION="http://alguna.direccion/cgi-bin/popup.exe">
Ingrese su información:<BR>
Nombre: <INPUT TYPE="text" NAME="nombre" SIZE=30 > <BR>
Email: <INPUT TYPE="text" NAME="email " SIZE=30 > <BR>
Comentarios.<BR>
<TEXTAREA NAME="caja" rows=5 cols=40></TEXTAREA> <BR>
<BR>
Que lenguaje prefiere?<BR>
<SELECT NAME="lenguaje">
<OPTION SELECTED> Turbo Pascal
<OPTION> Turbo Pascal
<OPTION> Delphi
<OPTION> Visual Basic
<OPTION> Smalltalk/V
<OPTION> Cobol
</SELECT> <BR>
<BR>
Que le pareció la guía? <BR>
<INPUT TYPE="radio" NAME="opin" VALUE="a">Mala <BR>
<INPUT TYPE="radio" NAME="opin" VALUE="b">Regular<BR>
<INPUT TYPE="radio" NAME="opin" VALUE="c" CHECKED>Buena<BR>
<INPUT TYPE="radio" NAME="opin" VALUE="d">Muy Buena<BR>
<INPUT TYPE="radio" NAME="opin" VALUE="d">Excelente!<BR>
```


<INPUT TYPE="checkbox" NAME="email" VALUE="si">

Marque esta casilla si quiere recibir información vía email.

<INPUT TYPE="submit" NAME="boton_env" VALUE="Enviar">

<INPUT TYPE="reset" NAME="boton_res" VALUE="Borrar">

</FORM>

Salida en Pantalla.