FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

A LOS ASISTENTES A LOS CURSOS

Las autoridades de la Facultad de Ingeniería, por conducto del jefe de la División de Educación Continua, otorgan una constancia de asistencia a quienes cumplan con los requisitos establecidos para cada curso.

El control de asistencia se llevará a cabo a través de la persona que le entregó las notas. Las inasistencias serán computadas por las autoridades de la División, con el fin de entregarle constancia solamente a los alumnos que tengan un mínimo de 80% de asistencias.

Pedimos a los asistentes recoger su constancia el día de la clausura. Estas se retendrán por el periodo de un año, pasado este tiempo la DECFI no se hará responsable de este documento.

Se recomienda a los asistentes participar activamente con sus ideas y experiencias, pues los cursos que ofrece la División están planeados para que los profesores expongan una tesis, pero sobre todo, para que coordinen las opiniones de todos los interesados, constituyendo verdaderos seminarios.

Es muy importante que todos los asistentes llenen y entreguen su hoja de inscripción al inicio del curso, información que servirá para integrar un directorio de asistentes, que se entregará oportunamente.

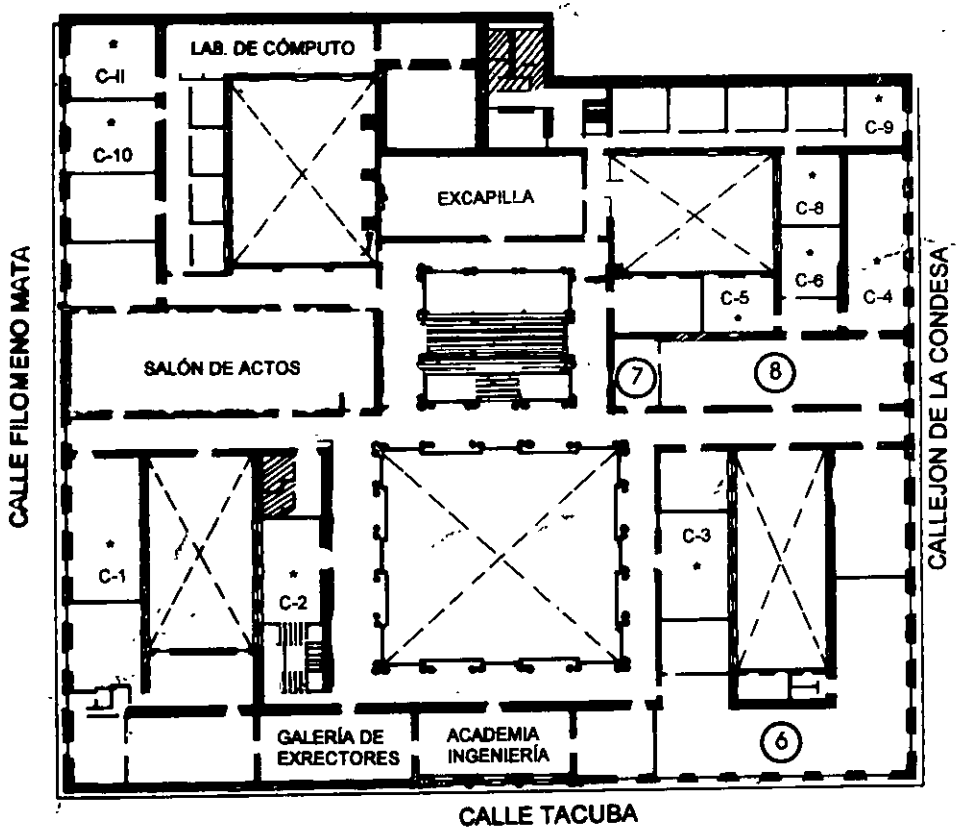
Con el objeto de mejorar los servicios que la División de Educación Continua ofrece, al final del curso deberán entregar la evaluación a través de un cuestionario diseñado para emitir juicios anónimos.

Se recomienda llenar dicha evaluación conforme los profesores impartan sus clases, a efecto de no llenar en la última sesión las evaluaciones y con esto sean más fehacientes sus apreciaciones.

Atentamente
División de Educación Continua.

Palacio de Minería, Calle de Tacuba No. 5, Primer piso, Delegación Cuauhtémoc, CP 06000, Centro Histórico, México D.F.,
APDO Postal M-2285 • Tels: 5521.4021 al 24, 5623.2910 y 5623.2971 • Fax: 5510.0573

PALACIO DE MINERIA



1er. PISO

GUÍA DE LOCALIZACIÓN

1. ACCESO
2. BIBLIOTECA HISTÓRICA
3. LIBRERÍA UNAM
4. CENTRO DE INFORMACIÓN Y DOCUMENTACIÓN "ING. BRUNO MASCANZONI"
5. PROGRAMA DE APOYO A LA TITULACIÓN
6. OFICINAS GENERALES
7. ENTREGA DE MATERIAL Y CONTROL DE ASISTENCIA
8. SALA DE DESCANSO

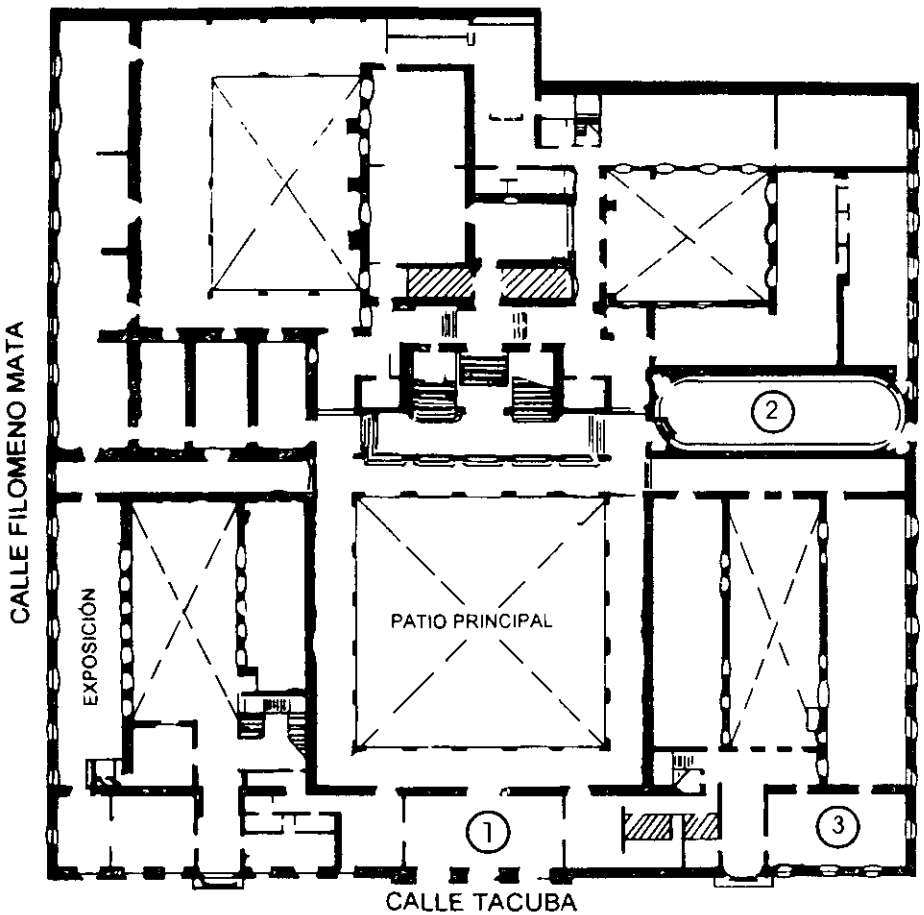
SANITARIOS

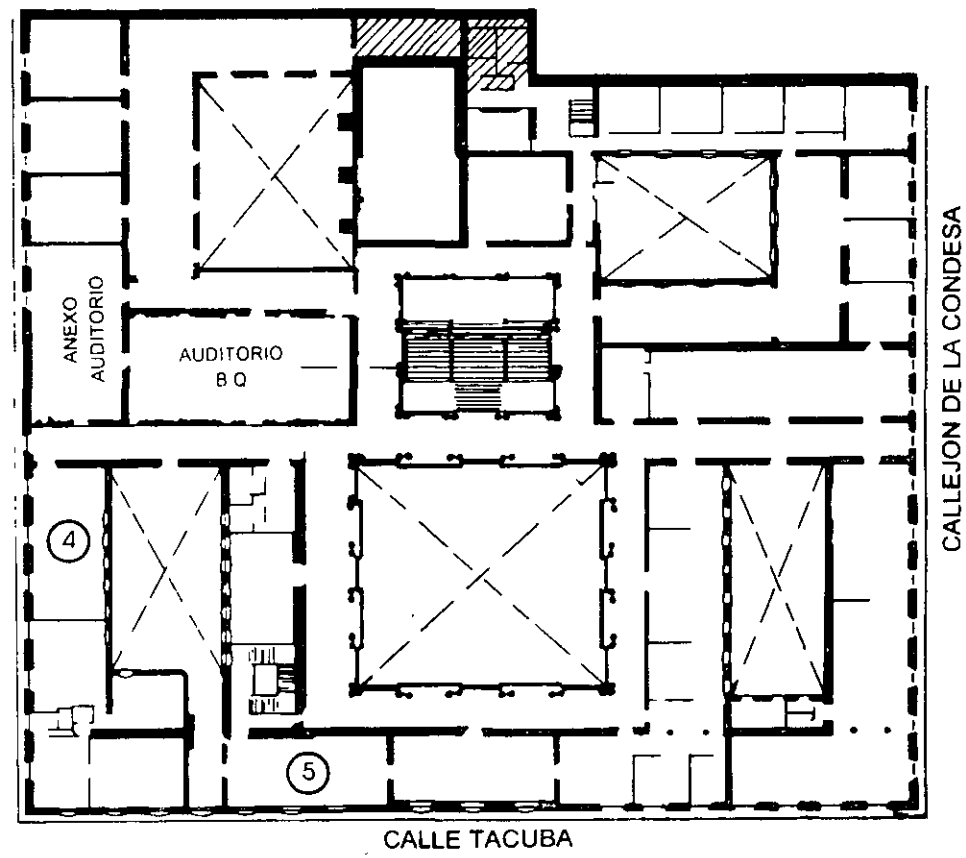
* AULAS

DIVISIÓN DE EDUCACIÓN CONTINUA
FACULTAD DE INGENIERÍA U.N.A.M.
CURSOS ABIERTOS

DIVISIÓN DE EDUCACIÓN CONTINUA
DEC

PALACIO DE MINERIA





PLANTA BAJA

MEZZANINNE

FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

Cursos Abiertos de Cómputo

Material Didáctico

ASP, PHP y bases de datos en sitios o portales WEB para no programadores

Septiembre, octubre 2005
CC53

Palacio de Minería, Calle de Tacuba #5, Centro Histórico, Delegación Cuauhtémoc, Código Postal 09440, México D.F.
APDO POSTAL M-2285 Teléfonos 5510-4673, 5623-2947 y 5623-2960

ASP, PHP Y BASES DE DATOS EN SITIOS O PORTALES WEB

PARA NO PROGRAMADORES

Coordinador Miguel Angel Flores
miguelangel.flores@gmail.com

Instructor: Mauricio Angulo
mauricioang@webmaestro.org.mx

Elección del servidor adecuado

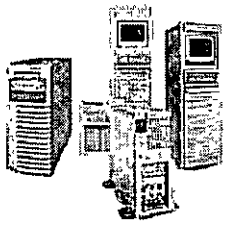
Existen básicamente dos grandes grupos bien diferenciados, los servidores basados en Windows NT (NT, 2000), y los basados en Unix (Linux, FreeBSD, Solaris, AIX, etc.).
La elección depende de las necesidades específicas del webmaster, y condicionará el crecimiento del sitio, por sus diferentes características y funciones.

Hay un temor a contratar servidores Unix por parte de los webmasters con conocimientos de Unix muy escasos o nulos, al desconocer con qué han de enfrentarse.
Gracias a los paneles de control que incorporan todos los planes de alojamiento, el sistema bajo el que trabaje el servidor es prácticamente transparente en su uso para la mayoría de aplicaciones.

Sólo en un uso avanzado puede necesitarse hacer uso del shell (línea de comandos) del sistema, si es que su plan de alojamiento lo permite. Tampoco afecta con qué sistema operativo se trabaje localmente, perfectamente puede manejarse un servidor Unix desde un equipo local Windows, son cosas independientes.

La información aquí publicada, debe considerarse sólo de orientación. Deberá siempre usted verificar la disponibilidad de los elementos que necesite en el plan de hosting concreto que esté examinando.

Servidor web



Pese a existir otras opciones, los servidores NT emplean típicamente IIS (Microsoft Internet Information Server) como servidor HTTP (web), y los Unix, emplean habitualmente Apache.

Los IIS, tienen una mayor integración con los sistemas de desarrollo y software de Microsoft, mientras los Apache, disponen de mayores opciones de configuración de forma relativamente sencilla, como son los redireccionamientos, personalización de páginas de error, accesos protegidos con contraseñas y otros detalles, además de existir una gran información sobre ellos en Internet.

Lenguajes de programación y scripts

Si bien hay lenguajes más habituales en alojamientos Unix que en Windows y al revés, hay disponibilidad de los siguientes lenguajes y scripts para ambos sistemas:

- **Perl:** El lenguaje para CGI por excelencia. Bajo Unix, tiene acceso a mayores partes del sistema, en Windows se encuentra parcialmente limitado.
- **PHP:** *Un buen lenguaje para webmasters principiantes, que se integra en el HTML.*
- **ASP:** *Para Unix, existe la versión ChiliSoft, pero si requiere de las últimas versiones, deberá decidirse por Windows.*
- **VBscript:**
- **Phyton**
- **Miva**
- **TCL**
- **C**
- **Java**
- **Jsp**
- **ColdFusion**

Bases de datos

El sistema de bases de datos a emplear es crítico, pues cuando se requieran, limitará los servidores a emplear. La migración de aplicaciones propias de uno a otro sistema es relativamente complejo, así como la migración del contenido de las bases de datos.

Existen numerosas aplicaciones que puede requerir en un momento dado, que le exigirán un sistema de bases de datos concreto. El habitualmente utilizado por aplicaciones para web, es MySQL, un potente sistema de bases de datos gratuito, que se ejecuta perfectamente en servidores Unix y NT y que también puede emplear localmente bajo Windows.

Base de datos	Windows NT	Unix	Comentarios
MySQL	Soportado	Soportado	Poco habitual en planes de alojamiento Windows. Sin coste de licencia.
MS Access	Soportado	No soportado	Sistema de base de datos empleado por el paquete Microsoft Office.
MS SQL	Soportado	No soportado	Sistema propietario de Microsoft.
mSQL	No soportado	Soportado	Gratuito para uso no comercial

Tanto en servidores Windows como Unix, existen posibilidades de conexiones ODBC con las bases de datos.

FrontPage

Si emplea FrontPage para el diseño de la web, ha de verificar que el servidor que desea contratar tiene instaladas las extensiones para FrontPage. Ambos sistemas las soportan y suelen formar parte del equipamiento incluido.

Seguridad

No hay ningún sistema 100% seguro. Durante mucho tiempo, los servidores IIS han protagonizado la mayor parte de incidentes. Algunos, como el gusano CodeRed, afectó a un alto número de ellos. Los incidentes de seguridad con servidores Apache, si bien han sufrido un incremento, son menos frecuentes.

Precios

En muchos casos, a igualdad de prestaciones, los alojamientos Unix se ofrecen a un menor precio. El coste de licencias de software -nulo en algunos casos- el menor requerimiento de hardware y mantenimiento, llevan a los sistemas Unix a una ventaja económica.

Conclusiones

Si sus necesidades le permiten elegir entre ambos sistemas y no se ha decidido todavía, la recomendación habitual es elegir un servidor Unix (habitualmente Linux). Son más estables y seguros, disponen normalmente de un mayor número de pequeñas opciones que no están disponibles en los Windows, hay mayor cantidad de software preparado para ellos, y en la red encontrará todo lo que necesite saber (y mucho más) sobre ellos.

Funcionamiento de un Web Site Dinámico

Para empezar

El funcionamiento de un Web-Site es un ejemplo típico de la arquitectura cliente-servidor, en donde múltiples clientes se conectan a un servidor (en algunos casos varios) en forma simultanea. En general el servidor depende de la instalación del site mientras que el cliente suele ser un browser, en general Netscape Navigator o Microsoft Explorer. Como en todo esquema cliente-servidor debe existir un protocolo que especifique de que forma se comunican e intercambian datos el cliente y el servidor, el protocolo utilizado en un web site es el protocolo HTTP que funciona "encapsulado" sobre el protocolo TCP/IP.

Páginas dinámicas vs HTML

A pesar de que las páginas dinámicas nos puedan en un principio limitar a causa de su mayor complejidad con respecto al HTML, todas las ventajas que nos ofrecen compensan con creces este esfuerzo inicial. No obstante, hay que ser consciente del posible interés que pueda tener para uno el lanzarse en esta aventura de aprender un nuevo lenguaje y volver a rediseñar su propio sitio. Si la página en la que estamos pensando o que queremos rediseñar es relativamente pequeña, no necesita estar al día continuamente sino que sus contenidos son perennes y no hemos previsto el pagar por mantenerla, el empleo de páginas dinámicas puede quedarse grande y resultar a todas luces improductivo.

Por el contrario, si el sitio es extenso y sus contenidos cambian rápidamente, nos interesa el automatizar en la medida de lo posible todas las tareas de tal forma que podamos gestionar su explotación de la manera más óptima. Para dejar más claro hasta que punto resulta útil utilizar páginas dinámicas lo mejor será ejemplificarlo a partir de un sitio web modelo.

Supongamos que hemos decidido realizar un portal de televisión donde una de las informaciones principales a proveer podría ser la programación semanal. Efectivamente, esta información suele ser dada por las televisiones con meses de antelación y podría ser muy fácilmente almacenada en una base de datos. Si trabajásemos con páginas HTML, tendríamos que construir **una página independiente para cada semana** en la cual introduciríamos "a mano" cada uno de los programas de cada una de las cadenas. Asimismo, cada semana nos tendríamos que acordar de descolgar la página de la semana pasada y colgar la de la anterior. Todo esto podría ser fácilmente resuelto mediante páginas dinámicas. En este caso, lo que haríamos **sería crear un programa (solo uno) que se encargaría de recoger de la base de datos** de la programación aquellos programas que son retransmitidos en las fechas que nos interesan y de confeccionar una página donde aparecerían ordenados por cadena y por hora de retransmisión. De este modo, podemos automatizar un proceso y desentendernos de un aspecto de la página por unos meses.

Este hecho lo podríamos aplicar a otras situaciones: podemos preparar el horóscopo de todos los días, las promociones de un sitio de e-comercio, etc.

Además, tampoco resultaría complicado el introducir una pequeña caja de búsqueda que nos permitiera dar rápidamente con el programa que queremos ver, saber a qué hora y en qué cadena se emite. Volviendo a nuestro portal de televisión, en él hay una sección en la cual presentamos todas las series actualmente emitidas con comentarios sobre ella, fotos, etc. Podríamos, en lugar de hacer una página HTML por serie, hacer una única página dinámica en contacto con una base de datos en la cual visualizamos las fotos y comentarios relativos a la serie que nos interesa. Asimismo, si lo que buscamos es modificar el formato del texto de dicha sección, podemos automatizar este proceso sin necesidad de cambiar a mano cada una de las

etiquetas font y sin hacer uso de la hojas de estilo las cuales no son reconocidas por la totalidad de los navegadores.

Ejecutando código del lado del servidor o cliente

El navegador es una especie de aplicación capaz de interpretar las órdenes recibidas en forma de código HTML fundamentalmente y convertirlas en las páginas que son el resultado de dicha orden.

Cuando nosotros provocamos un clic sobre un enlace hipertexto, en realidad lo que pasa es que establecemos una petición de un archivo HTML residente en el servidor (un servidor que se encuentra continuamente conectado a la red) el cual es enviado e interpretado por nuestro navegador (el cliente).

Sin embargo, si la página que pedimos no es un archivo HTML, el navegador es incapaz de interpretarla y lo único que es capaz de hacer es salvarla en forma de archivo. Es por ello que, si queremos emplear lenguajes accesorios para realizar un sitio web, es absolutamente necesario que sea el propio servidor quien los ejecute e interprete para luego enviarlos al cliente (navegador) en forma de archivo HTML totalmente legible por él.

De modo que, cuando provocamos un clic sobre un enlace a una pagina que contiene un script en un lenguaje comprensible únicamente por el servidor, lo que ocurre en realidad es que dicho script es ejecutado por el servidor y el resultado de esa ejecución da lugar a la generación de un archivo HTML que es enviado al cliente.

Así pues, podemos hablar de lenguajes de lado servidor que son aquellos lenguajes que son reconocidos, ejecutados e interpretados por el propio servidor y que se envían al cliente en un formato comprensible para él. Por otro lado, los lenguajes de lado cliente (entre los cuales no sólo se encuentra el HTML sino también el Java y el JavaScript los cuales son simplemente incluidos en el código HTML) son aquellos que pueden ser directamente "digeridos" por el navegador y no necesitan un pretratamiento.

Cada uno de estos tipos tiene por supuesto sus ventajas y sus inconvenientes. Así, por ejemplo, un lenguaje de lado cliente es totalmente independiente del servidor, lo cual permite que la página pueda ser albergada en cualquier sitio sin necesidad de pagar más ya que, por regla general, los servidores que aceptan páginas con scripts de lado servidor son en su mayoría de pago o sus prestaciones son muy limitadas. Inversamente, un lenguaje de lado servidor es independiente del cliente por lo que es mucho menos rígido respecto al cambio de un navegador a otro o respecto a las versiones del mismo. Por otra parte, los scripts son almacenados en el servidor quien los ejecuta y traduce a HTML por lo que permanecen ocultos para el cliente. Este hecho puede resultar a todas luces una forma legítima de proteger el trabajo intelectual realizado.

En el dominio de la red, los lenguajes de lado servidor más ampliamente utilizados para el desarrollo de páginas dinámicas son el ASP, PHP y PERL.

El ASP (Active Server Pages) es un lenguaje derivado del Visual Basic desarrollado por Microsoft. Evidentemente su empleo se realiza sobre plataformas funcionando bajo sistema Windows NT.

El PHP podría ser considerado como el lenguaje análogo al ASP utilizado en plataformas Unix y Linux.

Estos dos lenguajes resultan bastante útiles para la explotación de bases de datos y su aprendizaje resulta accesible para una persona profana de la programación. Cualquiera de ellos resultaría la opción ideal a la hora de hacer evolucionar un sitio web realizado en HTML.

Conceptos básicos de programación para no programadores

Antes de abordar en detalle las particularidades de estos lenguajes, es importante guardar en espíritu toda una serie de nociones básicas comunes. Estos aspectos son sin duda conocidos por aquellos que hayan programado alguna vez y pueden ser muy rápidamente asimilados por todos los que estén familiarizados con las matemáticas. Teniendo en cuenta esto, he querido acercar estos conceptos a cualquier persona proponiendo definiciones poco rigurosas y carentes de detalles pero que en contrapartida permiten ser digeridas con más facilidad.

Variable

Una variable consiste en un elemento al cual le damos un nombre y le atribuimos un determinado tipo de información. Las variables pueden ser consideradas como la base de la programación.

De este modo podríamos escribir en un lenguaje ficticio o pseudocódigo:

```
a="perro"
b="muerde"
```

La variable que nosotros llamamos "a" posee un elemento de información de tipo texto que es "perro". Asimismo, la variable "b" contiene el valor "muerde".
Podríamos definir una tercera variable que fuese la suma de estas dos:

```
c=a+b
```

Si introdujésemos una petición de impresión de esta variable en nuestro lenguaje ficticio:

```
imprimir(c)
```

El resultado podría ser:

```
perro muerde
```

Podríamos de la misma forma trabajar con variables que contuviesen números y construir nuestro programa:

```
a=3
b=4
c=a+b
imprimir(c)
```

El resultado de nuestro programa sería: 7

Funciónes y procedimientos

La función podría ser definida como un conjunto de instrucciones que permiten procesar las variables para obtener un resultado. Puede que esta definición resulte un poco vaga si no nos servimos de un ejemplo para ilustrarla.

Supongamos que queremos calcular el valor total de un pedido a partir de la simple suma de los precios de cada uno de los artículos. Podríamos definir una función suma en nuestro lenguaje ficticio:

```
definir funcion suma(art1,art2,art3)
suma=art1+art2+art3
imprimir(suma)
fin funcion
```

Este supuesto programa nos permitiría calcular la suma de tres elementos e imprimir el resultado en pantalla. Lo interesante de utilizar este tipo de funciones es que ellas nos permiten su utilización sistemática tantas veces como queramos sin necesidad de escribir las instrucciones tantas veces como veces queremos utilizarla. Por supuesto, podemos prescindir de esta declaración de función e introducir una línea del siguiente tipo:

```
imprimir(art1+art2+art3)
```

Evidentemente, cuanto más complicada sea la función y más a menudo la utilicemos en nuestros scripts más útil resulta definirlas. Esta función suma podría ser utilizada en cualquier lugar de nuestro script haciendo una llamada del siguiente tipo:

```
ejecuta suma(4,6,9)
```

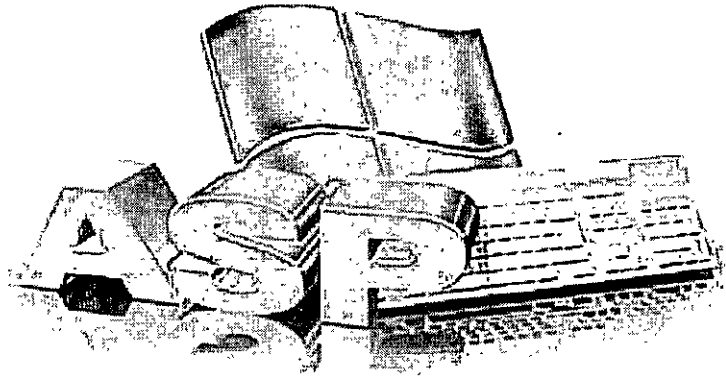
Cuyo resultado sería: **19**

Del mismo modo, los procedimientos son parecidos a las funciones. La diferencia consiste tan solo en que en estos últimos el interés no radica en el resultado obtenido sino más bien en las operaciones realizadas al ejecutarla (creación de un archivo, reenvío a otra página,...). En lenguajes como el PHP las funciones y los procedimientos son considerados como la misma cosa y para definirlos se hace usando los mismos comandos.

Tanto las variables como las funciones y los procedimientos deben ser nombradas sin servirse de acentos, espacios ni caracteres especiales para no correr riesgos de error .

Estos conceptos son básicos para una comprensión de la programación. No obstante, es posible que si es la primera vez que oímos hablar de ellos, su asimilación puede resultar parcial o nula. En realidad esto no es preocupante ya que a partir de los ejemplos de los capítulos siguientes y con la práctica de uno mismo se irán consolidando poco a poco.

Programación con ASP



Coordinador Miguel Angel Flores
migueangelflores@gmail.com

Instructor: Mauricio Angulo
mauricioang@webmaccers.org.mx

Introducción a la programación en ASP

Tal como hemos explicado, ASP (Active Server Pages) es la tecnología para la creación de páginas dinámicas del lado del servidor desarrollada por Microsoft.

El tipo de servidores que emplean este lenguaje son aquellos que funcionan con sistema operativo de la familia de Windows NT. Afortunadamente, también podemos visualizar páginas ASP sobre Windows 95/98, pero esto lo veremos más adelante.

Para escribir páginas ASP utilizamos un lenguaje de scripts, que se colocan en la misma página web junto con el código HTML. Comúnmente este lenguaje de scripts es Visual Basic Script, que deriva del conocido Visual Basic, aunque también se pueden escribir los scripts ASP en otro lenguaje: JScript, que deriva a su vez del conocido Javascript.

Existe una versión de Visual Basic Script en el lado cliente y otra en el lado del servidor. En los dos casos, como su nombre indica, el lenguaje de base es Visual Basic por lo que su aprendizaje puede ser perfectamente coordinado, ya que las sentencias y las sintaxis son prácticamente las mismas. En ASP, al estar programando páginas del lado del servidor, utlizaremos Visual Basic Script del lado del servidor y en este manual nos centraremos en este punto. El lector interesado por la sintaxis de Visual Basic Script y su programación del lado del cliente puede encontrar en este mismo sitio otro manual para tratar exclusivamente Visual Basic Script.

Este manual va destinado a aquellos que quieren comenzar de cero el aprendizaje de este lenguaje y que buscan en él la aplicación directa a su proyecto de sitio o a la mejora de su sitio HTML. Los capítulos son extremadamente simples, sino simplistas, buscando ser accesibles a la mayoría. Ellos serán complementados posteriormente con otros artículos de mayor nivel destinados a gente más experimentada.

Antes de comenzar a leer este manual es altamente aconsejable, sino imprescindible, haber leído previamente el manual sobre páginas dinámicas en el cual se explica a grandes rasgos qué es el ASP, algunos conceptos útiles sobre el modo de trabajar con páginas dinámicas al mismo tiempo que nos introduce algunos elementos básicos de la programación como pueden ser las variables y las funciones.

Del mismo modo, puede resultar extremadamente útil el haber leído o leer inmediatamente después el manual de Visual Basic Script en el cual se explica más en profundidad el lenguaje Visual Basic que resulta ser utilizado en la mayoría de scripts ASP.

Otra referencia a la cual haremos alusión es el tutorial de SQL que nos será de gran ayuda para el tratamiento de bases de datos.

Nuestra intención es la de ir publicando paulatinamente diferentes capítulos de modo que rogamos un poco de paciencia a aquellos que estén esperando la continuación. Todo irá llegando.

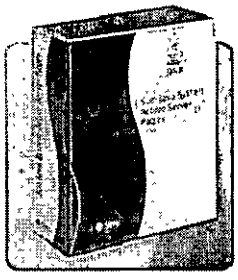
Esperamos que este manual resulte de vuestro agrado y que corresponda a nuestras expectativas: El poder acercar este lenguaje a todos aquellos amantes del desarrollo de webs que quieren dar el paso hacia las webs "profesionales".

Los scripts que usamos en estos primeros ejemplos pueden ser descargados aquí.

Si te interesa trabajar con un editor específico de ASP te recomendamos el MS Visual Interdev. Otra posibilidad es el Drumbeat de Macromedia aunque para empezar ninguno de los dos resulta absolutamente indispensable. También podemos elegir Homesite, un editor que no es específico para las ASP, pero que se comporta bastante bien y ofrece ayudas interesantes.

Instalación del Personal Web Server

En capítulos anteriores hemos explicado que, dada la naturaleza de los lenguajes de lado servidor, nos es imposible trabajar offline como hacíamos para el caso de las páginas HTML que almacenábamos en nuestro disco duro. También dijimos que esto no era completamente cierto ya que podíamos resolver este eventual problema instalándonos en nuestro PC un servidor propio.



Este servidor distribuido por Microsoft tiene dos versiones diferentes que son utilizadas dependiendo del equipo que estemos utilizando. Para los usuarios de **W95** o **W98**, la versión disponible se llama Personal Web Server (PWS). Si trabajamos bajo sistema **W-NT** el servidor a instalar es el Internet Information Server (IIS4). Existe también la posibilidad de trabajar en plataformas **UNIX** empleando en este caso el **ChilisoftASP**.

URL: **http://www.sun.com/software/chilisoft/**

Los usuarios de W95 tienen varias posibilidades para hacerse con el PWS: Descargarlo del sitio Microsoft , a partir de una antigua versión de Frontpage 98, instalándolo desde la opción pack de W-NT 4.0 o desde el CD de instalación de W98 (directorio add-ons/pws).

Los usuarios de Windows ME no tienen soporte para PWS, aunque se pueden probar una serie de pasos para lograr utilizarlo en el sistema. Este documento de Microsoft explica mejor este asunto.

URL: **http://support.microsoft.com/default.aspx?scid=kb;EN-US;q266456**

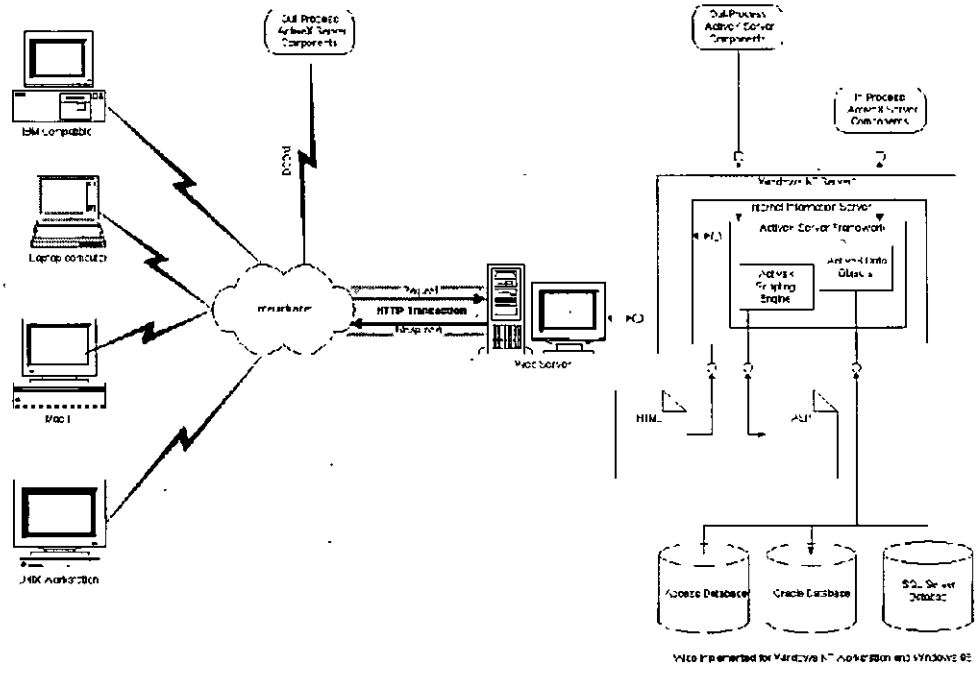
Por otro lado, los **usuarios de Windows 2000 deben utilizar IIS 5.0,** que se encuentra en la instalación. Es recomendable que lea también las notas de los visitantes al pie de página, porque encontraréis muchos más datos sobre problemas en distintas versiones y compatibilidades con otros sistemas que van apareciendo.

PWS podría ser considerado como una versión "light" del IIS4. En realidad en PWS no es suficientemente versátil para ejercer de servidor de un sitio de un tamaño mediano aunque si que podría en un momento dado hacerse cargo de un sitio de tamaño reducido y no muy concurrido. De todas formas, la utilidad del PWS radica sobre todo en que nos permite realizar las pruebas del sitio que vayamos a desarrollar en "local" sin necesidad de colgar nuestros archivos en el servidor que alberga nuestro sitio cada vez que queramos hacer una prueba sobre una pequeña modificación introducida. Esto resulta a todas luces práctico sobre todo para principiantes que necesitan hacer pruebas con una relativa frecuencia permitiendo el ahorro de mucho tiempo.

Conexión a BD

El siguiente paso, una vez instalado el servidor que nos permite trabajar en local, es crear los vínculos con las bases de datos que explotaremos en nuestros scripts. En efecto, la utilización de páginas dinámicas está muy frecuentemente asociada con el empleo de bases de datos.

Una base de datos es sencillamente un conjunto de tablas en las que almacenamos distintos registros (artículos de una tienda virtual, proveedores o clientes de una empresa, películas en cartelera en el cine, etc). Estos registros son catalogados en función de distintos parámetros que los caracterizan y que presentan una utilidad a la hora de clasificarlos. Así, por ejemplo, los artículos de una tienda virtual podrían catalogarse a partir de distintos campos como puede ser un número de referencia, nombre del artículo, descripción, precio, proveedor, etc.



Las bases de datos son construidas sirviéndose de aplicaciones tales como el **Microsoft Access o el MySQL las cuales resultan bastante sencillas de utilizar con unos conceptos mínimos.**

Nuestro objeto aquí no es explicar la forma de explotarlas sino cómo establecer una conexión entre la base de datos, almacenada en cualquier lugar del disco duro y nuestra página web alojada también en cualquier parte y reconocida por nuestro servidor personal a partir del directorio virtual.

Para crear este vínculo, nos servimos de los conectores ODBC (Open DataBase Connectivity) los cuales establecen el enlace con la base de datos.

El primer paso para crear esta conexión es ir al panel de configuración y abrir el icono ODBC 32bits. Dentro de él, deberemos crear un DSN (Data Source Name) de tipo sistema o usuario. Para ello nos colocamos en la solapa correspondiente (DSN sistema o DSN usuario) y seleccionamos

"Añadir". A continuación se nos pedirá seleccionar los controladores de la aplicación que hemos utilizado para crear la base de datos, el nombre que le queremos asignar (aquel que empleemos en nuestros scripts) y el camino para encontrarla en el disco duro. Esta DSN permite en realidad definir la base de datos que será interrogada sin necesidad de pasar por la aplicación que hayamos utilizado para construirla, es decir, con simples llamadas y órdenes desde nuestros archivos ASP podremos obtener los datos que buscamos sin necesidad de ejecutar el Access o el MySQL los cuales, evidentemente, no tendrán por qué encontrarse en el servidor donde trabajemos.

Inicio a la programación en ASP

Dado que el lenguaje ASP está muy frecuentemente embebido dentro del código HTML, es importante poder marcar al servidor qué partes están escritas en un lenguaje y cuáles en otro. Es por ello que todas las partes del archivo que están escritas en ASP estarán siempre delimitadas por los símbolos: <% y %>.

De este modo, cuando realicemos nuestros scripts, lo primero que debemos definir es el tipo de lenguaje utilizado, lo cual se hace del siguiente modo:

<% @ LANGUAGE="VBSCRIPT" %> Para el caso en el que programemos en Visual Basic Script

<% @ LANGUAGE="JSCRIPT" %> Si nos servimos del Java Script en servidor para programar en ASP

Los scripts que serán presentados en este manual estarán basados en el VBS, el cual presenta toda una serie de prestaciones que lo hacen sin duda más accesible y apto para ASP. No es por nada que es el propio Microsoft quien ha creado ambos.

Con los elementos que hemos presentado hasta ahora, ya estamos en situación de poder escribir nuestro primer programa en ASP. Vamos a crear un programa que calcule el 20% de impuestos que habría que añadir a una serie de artículos. Para plasmar el concepto de función, explicado en el manual de páginas dinámicas, vamos a definir una función "impuesto" que emplearemos sucesivas veces. El programa podría resultar algo así:

```
<% @ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Funcion impuesto</TITLE>
</HEAD>
<BODY>
<%Function impuesto(precio_articulo)
precio_final=precio_articulo+precio_articulo*20/100
Response.Write precio_final
End Function%>
Un libro de 3500 ptas. se quedará en un precio de <% impuesto(3500) %>
<br>
Una camisa de 6000 ptas. tendrá un precio final de <% impuesto(6000) %>
<br>
Un CD de música de 2000 ptas. costaría <% impuesto(2000) %> ptas.
</BODY>
</HTML>
```

Como puede verse, el script contiene dos partes fundamentales: Una primera en la que definimos la función que llamamos impuesto que depende únicamente de una variable (precio_articulo). Impuesto permite añadir un 20% al precio del artículo e imprimir el resultado en pantalla (Response.Write). En la segunda parte nos servimos de la función para realizar los cálculos necesarios y mostrarlos en pantalla acompañados de texto.

Resulta muy interesante, una vez ejecutado el script, ver el código fuente. Como puede verse, el código HTML que muestra el browser no coincide con el que nosotros hemos escrito. Algo que no debe sorprendernos ya que, como ya hemos explicado, el servidor se encarga de procesarlo y hacerlo comprensible al navegador.

Bucles y condiciones I

La programación exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución. Como ejemplo, podríamos hacer alusión a un script que ejecute una secuencia diferente en función del día de la semana en el que nos encontramos. Este tipo de acciones pueden ser llevadas a cabo gracias a una paleta de instrucciones presentes en la mayoría de los lenguajes. En este capítulo describiremos someramente algunas de ellas propuestas por el VBS y que resultan de evidente utilidad para el desarrollo de páginas ASP.

Para evitar el complicar el texto, nos limitaremos a introducir las más importantes dejando de lado otras cuantas que podrán ser fácilmente asimilables a partir de ejemplos prácticos. Para una mayor información sobre este tipo de elementos así como sobre los tipos de variables, referirse al manual de VBScript que trata más detenidamente estos aspectos.

Las condiciones: IF

Cuando queremos que el programa, llegado a un cierto punto, tome un camino determinado en determinados casos y otro diferente si las condiciones de ejecución difieren, nos servimos del conjunto de instrucciones If, Then y Else. La estructura de base de este tipo de instrucciones es la siguiente:

```
IF condición THEN
  Instrucción 1
  Instrucción 2
  ...
ELSE
  Instrucción A
  Instrucción B
  ...
END IF
```

Llegados a este punto, el programa verificará el cumplimiento o no de la condición. Si la condición es cierta las instrucciones 1 y 2 serán ejecutadas. De lo contrario (Else), las instrucciones A y B serán llevadas a cabo.

Una vez finalizada la estructura, deberemos cerrar con un End If.

Esta estructura de base puede complicarse un poco más si tenemos cuenta que no necesariamente todo es blanco o negro y que muchas posibilidades pueden darse. Es por ello que otras condiciones pueden plantearse dentro de la condición principal. Hablamos por lo tanto de condiciones anidadas que tendrían una estructura del siguiente tipo:

```
IF condición THEN
  Instrucción 1
  Instrucción 2
  ...
ELSE
  IF condición2 THEN
    Instrucción A
    Instrucción B
    ...
  ELSE
    Instrucción X
    ...
  END IF
END IF
```

De este modo podríamos introducir tantas condiciones como queramos dentro de una condición principal. En este tipo de estructuras es importante cerrar correctamente cada uno de los IF con sus END IF correspondientes. De gran ayuda es la instrucción ELSE IF que permite en una sola línea y sin necesidad de añadir un END IF introducir una condición anidada.

El uso de esta herramienta resultará claro con un poco de práctica. Pongamos un ejemplo sencillo de utilización de condiciones. El siguiente programa permitiría detectar la lengua empleada por el navegador y visualizar un mensaje en dicha lengua.

```
<% @ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Detector de Lengua</TITLE>
</HEAD>
<BODY>
<%
'Antes de nada introducimos mensajes en forma de variables
  espanol="Hola"
  ingles="Hello"
  frances="Bonjour"

'Ahora leemos del navegador cuál es su lengua oficial
  idioma=Left(Request.ServerVariables("HTTP_ACCEPT_LANGUAGE"),2)

'Formulamos las posibilidades que se pueden dar
If idioma="es" Then
  Response.Write espanol
ElseIf idioma="fr" Then
  Response.Write frances
Else
  Response.Write ingles
End If %>
</BODY>
</HTML>
```

Para poder ver el funcionamiento de este script es necesario cambiar el idioma preferido lo cual puede ser realizado a partir del menú de opciones del navegador.

Como puede verse, las variables que contienen texto son almacenadas entre comillas.

Para leer la lengua aceptada por el navegador lo que hacemos es definir una variable (idioma) que recoge las dos primeras letras empezando por la izquierda del idioma aceptado por el navegador ("HTTP_ACCEPT_LANGUAGE"). Este idioma aceptado puede ser requerido como una variable del objeto ServerVariables. Por el momento dejaremos esto tal cual, ya nos encargaremos de verlo más detenidamente en otros capítulos.

La tercera parte de script se encarga de ver si el navegador está en español (es), francés (fr) o en cualquier otro idioma que no sea ninguno de estos dos y de imprimir cada uno de los mensajes que proceda en cada caso.

Otro punto a comentar es el hecho de poder comentar los programas. Como puede observarse, dentro del script hemos introducido unos mensajes que nos sirven para leerlo más fácilmente. Estos mensajes no ejercen ninguna influencia en el desarrollo del mismo. Para introducirlos es necesario escribirlos detrás de un apóstrofe: '

Los comentarios son de gran utilidad cuando tratamos con programas muy extensos y complicados los cuales. En estos casos, resultan de gran ayuda a la hora de depurar fallos o introducir modificaciones. Es altamente aconsejable el acostumbrarse a utilizarlos.

Bucles y condiciones II

Los bucles FOR

En muchas ocasiones resulta necesario el ejecutar un conjunto de instrucciones un número definido de veces. Esto puede ser llevado a cabo a partir de la instrucción FOR/NEXT.

La estructura clásica:

```
FOR contador=número inicial to número final STEP incremento
  Instrucción 1
  Instrucción 2
  ...
NEXT
```

A partir de este tipo de estructuras ejecutamos las instrucciones contenidas entre el FOR y el NEXT un cierto número de veces definido por el número inicial, final y el incremento. El incremento resulta de 1 por defecto.

Pongamos un ejemplo:

```
<% ... LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Bucle for/next</TITLE>
</HEAD>
<BODY>

<%For i=1 to 5%>
<font size=<%Response.Write i%>>Vuelta número <%Response.Write i%></font><br>
<%Next

For i=5 to 1 Step -1%>
<font size=<%Response.Write i%>>Contamos atrás: <%Response.Write i%></font><br>
<%Next%>

</BODY>
</HTML>
```

Este script compuesto de dos bucles cuenta primero de 1 a 5. La variable i toma por lo tanto todos los valores enteros comprendidos entre estos dos números y puede ser utilizada dentro del bucle como lo hacemos en este caso para aumentar el tamaño de la letra. El segundo bucle realiza el proceso inverso (el incremento es negativo) produciendo una disminución del tamaño de la letra.

Lo que puede resultar interesante para ver hasta qué punto el programar páginas dinámicas puede ahorrarnos texto con respecto a la misma página programada en código HTML, es mirar el código fuente de la página a partir del navegador.

Bucles y condiciones III

Los bucles DO WHILE/LOOP

Otra forma de realizar este tipo de secuencias bucle es a partir de la instrucción DO WHILE. En este caso lo que especificamos para fijar la extensión del bucle no es el número de vueltas sino el que se cumpla o no una condición. La estructura de este tipo de bucles es análoga a la de los bucles FOR/NEXT:

```
DO WHILE condición
  Instrucción 1
  Instrucción 2
  ...
LOOP
```

El bucle se dará mientras la condición propuesta siga siendo válida.
Como se verá en ejemplos posteriores, este tipo de bucles resulta muy práctico para la lectura de bases de datos.

Todo este tipo de controladores de flujo (condiciones y bucles) pueden ser matizados y optimizados a partir del uso de operadores lógicos. Así, podemos exigir que sean dos las condiciones que se den para llevar a cabo un conjunto de instrucciones:

```
IF condición 1 AND condición 2 THEN ...
```

También podemos requerir que sea una de las dos:

```
IF condición 1 OR condición 2 THEN...
```

Del mismo modo, es posible exigir que la condición de un bucle DO sea la inversa a la enunciada:

```
DO WHILE NOT condición
```

He aquí, en conclusión, un conjunto de recursos básicos para controlar el desarrollo de programas. Su utilidad resultará más que patente y su uso irá haciéndose intuitivo a medida que nos familiaricemos con el lenguaje.

Los objetos ASP

El ASP es un lenguaje diseñado para la creación de aplicaciones en internet. Esto quiere decir que existen toda una serie de tareas bastante corrientes a las cuales debe dar un tratamiento fácil y eficaz. Nos referimos por ejemplo al envío de e-mails, acceso a archivos, gestión de variables del cliente o servidor como pueden ser su IP o la lengua aceptada...

El lenguaje VB propiamente dicho no da una solución fácil y directa a estas tareas sino que invoca a los denominados objetos que no son más que unos módulos incorporados al lenguaje que permiten el desarrollo de tareas específicas. Estos objetos realizan de una manera sencilla toda una serie de acciones de una complejidad relevante. A partir de una llamada al objeto este realizará la tarea requerida. En cierta forma, estos objetos nos ahorran el tener que hacer largos programas para operaciones sencillas y habituales.

Algunos de estos objetos están incorporados en el propio ASP, otros deben de ser incorporados como si se tratase de componentes accesorios. Por supuesto, no podríamos ejecutar correctamente un script en el cual tuviésemos que llamar a un objeto que no estuviese integrado en el servidor. Este tipo de "plug-in" son generalmente comprados por el servidor a empresas que los desarrollan.

Como todo objeto del mundo real, los objetos del mundo informático tienen sus propiedades que los definen, realizan un cierto numero de funciones o métodos y son capaces de responder de una forma definible ante ciertos eventos.

Dado el nivel de esta obra, la descripción de la totalidad de objetos con sus métodos y propiedades resulta ciertamente fuera de lugar. Nos contentaremos pues con ir describiendo los más frecuentemente utilizados y ejemplificarlos de la manera más práctica dejando la enumeración exhaustiva en forma de apéndice.

Objeto Request I

Bucles y condiciones son muy útiles para procesar los datos dentro de un mismo script. Sin embargo, en un sitio internet, las páginas vistas y los scripts utilizados son numerosos. Muy a menudo necesitamos que nuestros distintos scripts estén conectados unos con otros y que se sirvan de variables comunes. Por otro lado, el usuario interacciona por medio de formularios cuyos campos han de ser procesados para poder dar una respuesta. Todo este tipo de factores dinámicos han de ser eficazmente regulados por un lenguaje como el ASP.

Como veremos, todo este tipo de aspectos interactivos pueden ser gestionados a partir del objeto Request.

El objeto Request nos devuelve informaciones del usuario que han sido enviadas por medio de formularios, por URL o a partir de cookies (veremos de qué se tratan seguidamente). También nos informa sobre el estado de ciertas variables del sistema como pueden ser la lengua utilizada por el navegador, el número IP del cliente.

Transferir variables por URL

Para pasar las variables de una página a otra lo podemos hacer introduciendo dicha variable en la dirección URL de la página destino dentro del enlace hipertexto. La sintaxis sería la siguiente:

«a href="destino.asp?variable1=valor1&variable2=valor2&..."»« /a»

Para recoger la variable en la página destino lo hacemos por medio del objeto Request con el método Querystring:

```
Request.querystring("variable1")
Request.querystring("variable2")
```

Las dos páginas serían así:

```
<HTML>
<HEAD>
<TITLE>Página origen.asp</TITLE>
</HEAD>
<BODY>
<a href="destino.asp?saludo=hola&texto=Esto es una variable texto">Paso
variables saludo y texto a la página destino.asp</a>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>destino.asp</TITLE>
</HEAD>
<BODY>
Variable saludo: <%Response.Write Request.Querystring("saludo")%> <br>
Variable texto: <%Response.Write Request.Querystring("texto")%><br>
</BODY>
</HTML>
```

Objeto Request II
Transferir variables por formulario

El proceso es similar al explicado para las URLs. Primeramente, presentamos una primera página con el formulario a rellenar y las variables son recogidas en una segunda página que las procesa:

```
<HTML>
<HEAD>
<TITLE>formulario.asp</TITLE>
</HEAD>
<BODY>
<FORM METHOD="POST" ACTION="destino2.asp">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"><br>
Apellidos<br>
<INPUT TYPE="TEXT" NAME="apellidos"> <br>
<INPUT TYPE="SUBMIT">
</FORM>
</BODY>
</HTML>


<HTML>
<HEAD>
<TITLE>destino2.asp</TITLE>
</HEAD>
<BODY>
Variable nombre: <%=Request.Form("nombre")%><br>
Variable apellidos: <%=Request.Form("apellidos")%>
</BODY>
</HTML>
```

Otras utilidades de Request: las ServerVariables

El objeto Request nos da acceso a otras informaciones relativas al cliente y el servidor las cuales pueden resultar de una gran utilidad. Estas informaciones son almacenadas como variables las cuales son agrupadas en una colección llamada ServerVariables.

Dentro de esta colección tenemos variables tan interesantes como:

HTTP_ACCEPT_LANGUAGE	Nos informa de la lengua preferida por el navegador
HTTP_USER_AGENT	Indica cuál es el navegador utilizado.
PATH_TRANSLATED	Nos devuelve el path físico del disco duro del servidor en el que se encuentra nuestro script
SERVER_SOFTWARE	Nos dice qué tipo de software utiliza el servidor

Para visualizar en pantalla alguna de estas variables, debemos escribir algo como:

```
Response.write request.servervariables("nombre de la variable")
```

Una forma rápida de visualizar todas estas variables es a partir de un script con esta secuencia:

```
<%
For Each elemento in Request.ServerVariables
   Response.Write elemento& ": "&Request.ServerVariables(elemento)& "<br>"
Next
%>
```

Esto nos daría por un lado el nombre de la variable y del otro su valor. Este tipo de bucle For Each/Next se parece a otros ya vistos. En este caso, el bucle se realiza tantas veces como elementos tiene la colección (ServerVariables) que no es más que el conjunto de elementos comprendidos en la extensión del objeto (Request). Este tipo de bucle es aplicable a otras colecciones de éste y otros objetos como por ejemplo los Request.Form o Request.Querystring o las cookies. De esta forma seríamos capaces de visualizar el nombre y contenido de tales colecciones sin necesidad de enunciarlas una por una.

Objeto Response

Tal y como se ha visto, el objeto Request gestiona todo lo relativo a entrada de datos al script por parte del usuario (formularios), provenientes de otra URL, del propio servidor o del browser.

Nos queda pues por explicar cómo el script puede "responder" a estos estímulos, es decir, cómo, después de procesar los debidamente los datos, podemos imprimir estos en pantalla, inscribirlos en las cookies o enviar al internauta a una u otra pagina. En definitiva, queda por definir la forma en la que ASP regula el contenido que es enviado al navegador.

Esta función es tomada en cargo por el objeto Response el cual ha sido ligeramente mencionado en capítulos precedentes concretamente en asociación al método Write.

En efecto, sentencias del tipo:

```
<%Response.Write "Una cadena de texto"%>
```

o bien,

```
<%Response.Write variable%>
```

Tienen como cometido imprimir en el documento HTML generado un mensaje o valor de variable. Este método es tan comúnmente utilizado que existe una abreviación del mismo de manera a facilitar su escritura:

```
<% = variable %> es análogo a <%response.write variable%>
```

Es importante precisar el hecho que imprimir en el documento HTML no significa necesariamente visualizar en pantalla ya que podríamos servirnos de estas etiquetas para crear determinadas etiquetas HTML. He aquí un ejemplo de lo que se pretende decir:

```
<% path="http://www.misitio.com/graficos/imagen.gif" %>

<img src="<% =path %>">
```

Este fragmento de script nos generaría un código HTML que sería recibido en el navegador del siguiente modo:

```
<img src="http://www.misitio.com/graficos/imagen.gif">
```

Otro elemento interesante de este objeto Response es el método Redirect. Este método nos permite el enviar automáticamente al internauta a una página que nosotros decidamos.

Esta función, a todas luces práctica, puede ser empleada en scripts en los que enviamos al visitante de nuestro sitio a una u otra página en función del navegador que utiliza o la lengua que prefiere. También es muy útil para realizar páginas "escondidas" que realizan una determinada función sin mostrar ni texto ni imágenes y nos envían a continuación a otra página que es en realidad la que nosotros recibimos en el navegador.

Aquí se puede ver una secuencia de script que nos permitiría usar este método para el caso en el que tengamos páginas diferentes para distintas versiones de navegadores. En ella emplearemos un objeto que nos debe parecer familiar ya que lo acabamos de ver (Request.ServerVariables):

```
<%
tipo_navegador = Request.ServerVariables("HTTP_USER_AGENT")
If Instr(1, tipo_navegador, "MSIE 3", 1) <> 0 then
Response.Redirect "MSIE3.asp"
Elseif Instr(1, tipo_navegador, "MSIE 4", 1) <> 0 then
Response.Redirect "MSIE4.asp"
Elseif Instr(1, tipo_navegador, "MSIE 5", 1) <> 0 then
Response.Redirect "MSIE5.asp"
Elseif Instr(1, tipo_navegador, "Mozilla/4", 1) <> 0 then
Response.Redirect "Netscape4.asp"
Else
Response.Redirect "cualquiera.asp"
%>
```

Por supuesto, acompañando este script, debemos tener los correspondientes archivos MSIE3.asp, MSIE4.asp, MSIE5.asp... Cada uno de ellos con las particularidades necesarias para la buena visualización de nuestra página en tales navegadores.

Las famosas cookies

Sin duda este término resultara familiar para muchos. Algunos lo habrán leído u oído pero no saben de qué se trata. Otros sin embargo sabrán que las cookies son unas informaciones almacenadas por un sitio web en el disco duro del usuario. Esta información es almacenada en un archivo tipo texto que se guarda cuando el navegador accede al sitio web.

Es posible, por supuesto, ver estos archivos. Para abrirlos hay que ir al directorio C:\Windows\Cookies para los usuarios de IE 4+ o a C:\...\Netscape\Users\defaultuser para usuarios de Netscape. Como podréis comprobar, en la mayoría de los casos la información que se puede obtener es indescifrable.

La utilidad principal de las cookies es la de poder identificar al navegador una vez éste visita el sitio por segunda vez y así, en función del perfil del cliente dado en su primera visita, el sitio puede adaptarse dinámicamente a sus preferencias (lengua utilizada, colores de pantalla, formularios rellenados total o parcialmente, redirección a determinadas páginas...).

Para crear un archivo cookies, modificar o generar una nueva cookie lo podemos hacer a partir del objeto Response con una sintaxis como la siguiente:

```
Response.Cookies("nombre de la cookie") = valor de la cookie
```

El valor de la cookie puede ser una variable, un numero o una cadena delimitada por comillas.

Es importante saber que las cookies tienen una duración igual a la sesión, es decir, a menos que lo especifiquemos, el archivo texto generado se borrará desde el momento en que el usuario haya abandonado el sitio por un tiempo prolongado (generalmente unos 20 minutos) o que el navegador haya sido cerrado. Podemos arreglar este supuesto inconveniente mediante la propiedad Expires:

```
Response.Cookies("nombre de la cookie").expires = #01/01/2002#
```

Esto nos permite decidir cual es la fecha de caducidad de la cookie. Hay que tener en cuenta que esto no es más que hipotético ya que el usuario es libre de borrar el archivo texto cuando le plazca.

Por otra parte, es interesante señalar que el hecho de que definir una cookie ya existente implica el borrado de la antigua. Del mismo modo, el crear una primera cookie conlleva la generación automática del archivo texto.

Para leer las cookies, nada más fácil que usando el objeto Request de la siguiente forma:

```
variable = Request.Cookies("nombre de la cookie")
```

Si por ejemplo quisiéramos recuperar del archivo txt la cookie correspondiente a la lengua del cliente y almacenarlo en nuestro script para futuros usos, podríamos escribir:

```
lengua=Request.Cookies("lengua")
```

Las cookies son una herramienta fantástica para personalizar nuestra página pero hay que ser cautos ya que, por una parte, no todos los navegadores las aceptan y por otra, se puede deliberadamente impedir al navegador la creación de cookies. Es por ello que resultan un complemento y no una fuente de variables infalible para nuestro sitio.

Objeto Session

En los programas que hemos visto hasta ahora, hemos utilizado variables que solo existían en el archivo que era ejecutado. Cuando cargábamos otra página distinta, los valores de estas variables se perdían a menos que nos tomásemos la molestia de pasarlos por la URL o inscribirlos en las cookies o en un formulario para su posterior explotación. Estos métodos, aunque útiles, no son todo lo prácticos que podrían en determinados casos en los que la variable que queremos conservar ha de ser utilizada en varios scripts diferentes y distantes los unos de los otros.

Podríamos pensar que ese problema puede quedar resuelto con las cookies ya que se trata de variables que pueden ser invocadas en cualquier momento. El problema, ya lo hemos dicho, es que las cookies no son aceptadas ni por la totalidad de los usuarios ni por la totalidad de los navegadores lo cual implica que una aplicación que se sirviera de las cookies para pasar variables de un archivo a otro no sería 100% infalible.

Nos resulta pues necesario el poder declarar ciertas variables que puedan ser reutilizadas tantas veces como queramos dentro de una misma sesión. Imaginemos un sitio multilingüe en el que cada vez que queremos imprimir un mensaje en cualquier página necesitamos saber en qué idioma debe hacerse. Podríamos introducir un script identificador de la lengua del navegador en cada uno de los archivos o bien declarar una variable que fuese valida para toda la sesión y que tuviese como valor el idioma reconocido en un primer momento.

Estas variables que son válidas durante una sesión y que luego son "olvidadas" son definidas con el objeto Session de la siguiente forma:

```
Session("nombre de la variable") = valor de la variable
```

Una vez definida, la variable Session, será almacenada en memoria y podrá ser empleada en cualquier script del sitio web.

La duración de una sesión viene definida por defecto en 20 minutos. Esto quiere decir que si en 20 minutos no realizamos ninguna acción, el servidor dará por finalizada la sesión y todas las variables Session serán abandonadas. Esta duración puede ser modificada con la propiedad Timeout:

```
Session.Timeout = n° de minutos que queramos que dure
```

Una forma de borrar las variables Session sin necesidad de esperara que pase este plazo es a partir del método Abandon:

```
Session.Abandon
```

De este modo todas las variables Session serán borradas y la sesión se dará por finalizada. Este método puede resultar practico cuando estemos haciendo pruebas con el script y necesitemos reinicializar las variables.

Lo que se suele hacer es crear un archivo en el que se borran las cookies y se abandona la sesión. Este archivo será ejecutado cuando queramos hacer borrón y cuenta nueva:

```
<%@ LANGUAGE="VBSCRIPT" %>
<HTML>
<HEAD>
<TITLE>Puesta a cero</TITLE>
</HEAD>
<BODY>
<%
For Each galleta in Response.Cookies
  Galleta=""
Next
Session.Abandon
%>
Borrón y cuenta nueva!!<br>
<a href="url de la página de inicio">Volver al principio</a>
</BODY>
</HTML>
```

Selecciones en una tabla

Dentro de una base de datos, organizada por tablas, la selección de una tabla entera o de un cierto numero de registros resulta una operación rutinaria.

A partir de esta selección se puede posteriormente efectuar toda una serie de cambios o bien realizar una simple lectura.

El siguiente script nos permite realizar la lectura de la tabla clientes contenida en nuestra base de datos. A primera vista todo nos puede parecer un poco complejo, pero nada más lejos de la realidad.

```
<HTML>
<HEAD>
<TITLE>Lectura de registros de una tabla</TITLE>
</HEAD>
<BODY>
<h1><div align="center">Lectura de la tabla</div></h1>
<br>
<br>
<%
'Antes de nada hay que instanciar el objeto Connection
Set Conn = Server.CreateObject("ADODB.Connection")

'Una vez instanciado Connection lo podemos abrir y le asignamos la base de datos donde vamos a efectuar
las operaciones
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL que nos servira para hablar a la BD
sSQL="Select * From Clientes Order By nombre"

'Ejecutamos la orden
set RS = Conn.Execute(sSQL)

'Mostramos los registros%>
<table align="center">
<tr>
```

```
<th>Nombre</th>
<th>Teléfono</th>
</tr>
<%
Do While Not RS.Eof
%>
<tr>
<td><%=RS("nombre")%></td>
<td><%=RS("telefono")%></td>
</tr>
<%
RS.MoveNext
Loop

'Cerramos el sistema de conexion
Conn.Close
%>

</table>

<div align="center">
<a href="insertar.html">Añadir un nuevo registro</a><br>
<a href="actualizar1.asp">Actualizar un registro existente</a><br>
<a href="borrar1.asp">Borrar un registro</a><br>
</div>

</BODY>
</HTML>
```

Antes de nada, si lo que deseamos es interrogar una base de datos, lo primero que hay que hacer es obviamente establecer la conexión con ella. Esto se hace a partir del objeto Connection el cual es "invocado" o más técnicamente dicho instanciado por medio de la primera instrucción en la cual el objeto toma el nombre arbitrario de la variable Conn .

El paso siguiente es abrir el objeto y asignarle la base de datos con la que debe contactar. En este caso, la base la hemos llamado Mibase. Este debe de ser el mismo nombre con el que la hemos bautizado cuando hemos configurado los conectores ODCB, además, este nombre no tiene por qué coincidir necesariamente con el nombre del archivo.

Una vez creada la conexión a nuestra base de datos, el paso siguiente es hacer nuestra petición. Esta petición puede ser formulada primeramente y almacenada en una variable (sSQL) para, a continuación, ser ejecutada por medio de la instrucción siguiente.

La petición que hemos realizado en este caso es la de seleccionar todos los campos que hay en la tabla clientes (* es un comodín) y ordenar los resultados por orden alfabético con respecto al campo nombre. Podemos ver más detalladamente este tipo de instrucciones SQL en nuestro manual de SQL.

El resultado de nuestra selección es almacenado en la variable RS en forma de tabla. Para ver la tabla lo que hay que hacer ahora es "pasearse" por esta tabla "virtual" RS la cual posee una especie de cursor que, a menos que se especifique otra cosa, apunta al primer registro de la selección. El objetivo ahora es hacer desplazarse al cursor a lo largo de la tabla para poder leerla en su totalidad. La forma de hacerlo es a partir de un bucle Do While el cual ya ha sido explicado anteriormente y que lo único que hace es ejecutar las instrucciones comprendidas entre el Do y el Loop siempre que la condición propuesta (Not RS.Eof) sea verdadera. Esto se traduce como "Ejecutar este conjunto de instrucciones mientras que la tabla de resultados (RS) no llegue al

final (Eof, End of File).

Las instrucciones incluidas en el bucle son, por un lado, la impresión en el documento de los valores de determinados campos (=RS("nombre del campo")) y por otro, saltar de un registro al otro mediante la instrucción RS.MoveNext.

Todo este conjunto de instrucciones ASP viene en combinación con un código HTML que permite su visualización en forma de tabla. Además, se han incluido unos enlaces que apuntan hacia otra serie de scripts que veremos más adelante y que formaran en conjunto una aplicación.

Es interesante ver el código fuente resultante de este script. En este caso el código que ve el cliente resulta sensiblemente más sencillo.

Creación de un nuevo registro

En este caso lo que buscamos es crear, a partir de los datos recibidos de un formulario, un nuevo registro en nuestra tabla clientes. Tendremos pues dos archivos diferentes, uno que podría ser un HTML puro en el que introducimos el formulario a rellenar y que nos envía al segundo, un script muy parecido al previamente visto para realizar una selección. He aquí los dos scripts:

```
<HTML>
<HEAD>
<TITLE>Insertar.html</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Insertar un registro</h1>
<br>
<FORM METHOD="POST" ACTION="Insertar.asp">
Nombre<br>
<INPUT TYPE="TEXT" NAME="nombre"> <br>
Teléfono<br>
<INPUT TYPE="TEXT" NAME="telefono"> <br>
<INPUT TYPE="SUBMIT" value="Insertar">
</FORM>
</div>
</BODY>
</HTML>
```

```
<HTML>
<HEAD>
<TITLE>Insertar.asp</TITLE>
</HEAD>
<BODY>

<%
'Recogemos los valores del formulario
nombre=Request.Form("nombre")
telefono= Request.Form("telefono")

'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL
sSQL="Insert Into Clientes (nombre, telefono) values (" & nombre & "," & telefono & ")"
```

```
'Ejecutamos la orden
set RS = Conn.Execute(sSQL)
%>

<h1> <div align="center">Registro insertado</div> </h1>
<div align="center"> <a href="lectura.asp">Visualizar el contenido de la base</a></div>

<%
'Cerramos el sistema de conexion
Conn.Close
%>

</BODY>
</HTML>
```

Como puede verse, la forma de operar es idéntica a la vista anteriormente para el display de una tabla. En este caso hemos introducido un enlace a este primer script de lectura para ver cómo los cambios se han hecho efectivos.

La construcción de la sentencia SQL se hace por fusión de los distintos elementos constitutivos. La forma de fusionarlos mediante el símbolo &. Todo lo que sea texto tiene que ir entre comillas. Sería interesante introducir una línea suplementaria en vuestro código para imprimir la sSQL formada. La línea sería del siguiente tipo:

```
Response.Write sSQL
```

Esta línea iría situada evidentemente después de haber construido la sentencia.

Actualización de un registro existente

Para mostrar cómo se actualiza un registro presente en nuestra base de datos, vamos a hacerlo a partir de un caso un poco más complejo para que empecemos a familiarizarnos con estas operaciones. Realizaremos un par de scripts que permitan cambiar el numero de teléfono de las distintas personas presentes en nuestra base. El nombre de estas personas, así como el nuevo numero de teléfono, serán recogidos por medio de un formulario.

El archivo del formulario va a ser esta vez un script ASP en el que efectuaremos una llamada a nuestra base de datos para construir un menú desplegable donde aparezcan todos los nombres. La cosa quedaría así:

```
<HTML>
<HEAD>
<TITLE> Actualizar1.asp</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Actualizar un registro</h1>
<br>

<%
'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"
%>

<FORM METHOD="POST" ACTION="actualizar2.asp">
```

```
Nombre<br>
<%
'Creamos la sentencia SQL y la ejecutamos
sSQL="Select nombre From clientes Order By nombre"
set RS = Conn.Execute(sSQL)
%>
<select name="nombre">
<%
'Generamos el menu desplegable
Do While not RS.eof%>
   <option><%=RS("nombre")%>
   <%RS.movenext
Loop
%>
</select>
<br>
Telefono<br>
<INPUT TYPE="TEXT" NAME="telefono"><br>
<INPUT TYPE="SUBMIT" value="Actualizar">
</FORM>
</div>

</BODY>
</HTML>
```

La manera de operar para construir el menú desplegable es la misma que para visualizar la tabla. De nuevo empleamos un bucle Do While que nos permite mostrar cada una de las opciones. El script de actualización será muy parecido al de inserción:

```
<TITLE>Actualizar2.asp</TITLE>
</HEAD>
<BODY>

<%
Recogemos los valores del formulario
nombre=Request.Form("nombre")
telefono= Request.Form("telefono")

Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL
sSQL="Update Clientes Set telefono='" & telefono & "' Where nombre='" & nombre & "'"

Ejecutamos la orden
set RS = Conn.Execute(sSQL)
%>

<h1><div align="center">Registro Actualizado</div></h1>
<div align="center"><a href="lectura.asp">Visualizar el contenido de la base</a></div>

<%
Cerramos el sistema de conexion
Conn.Close
%>

</BODY>
```

```
</HTML>
```

Nada que comentar al respecto salvo la estructura de la sentencia SQL que en este caso realiza un Update en lugar de un Insert. Aconsejamos, como para el caso precedente imprimir el valor de sSQL de manera a ver cómo queda la sentencia una vez construida.

Borrado de un registro

Otra de las operaciones elementales que se pueden realizar sobre una base de datos es el borrar un registro. Para hacerlo, SQL nos propone sentencias del tipo Delete. Veámoslo con un ejemplo aplicado a nuestra agenda. Primero, crearemos un menú desplegable dinámico como para el caso de las actualizaciones:

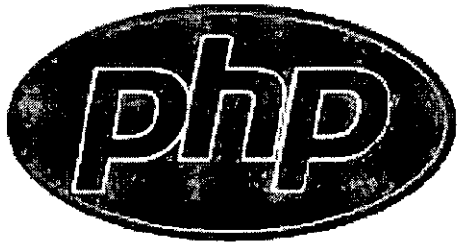
```
<HTML>
<HEAD>
<TITLE>Borrar1.asp</TITLE>
</HEAD>
<BODY>
<div align="center">
<h1>Borrar un registro</h1>
<br>
<%
'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"
%>

<FORM METHOD="POST" ACTION="borrar2.asp">
Nombre<br>
<%
'Creamos la sentencia SQL y la ejecutamos
sSQL="Select nombre From clientes Order By nombre"
set RS = conn.execute(sSQL)
%>
<select name="nombre">
<%
'Generamos el menu desplegable
Do While not RS.eof%>
  <option><%=RS("nombre")%>
  <%RS.movenext
Loop
%>
</select>
<br>
<INPUT TYPE="SUBMIT" value="Borrar">
</FORM>
</div>

</BODY>
</HTML>
```

El siguiente paso es hacer efectiva la operación a partir de la ejecución de la sentencia SQL que construimos a partir de los datos del formulario:

```
<HTML>
<HEAD>
<TITLE>Borrar2.asp</TITLE>
</HEAD>
<BODY>
<%
'Recogemos los valores del formulario
nombre=Request.Form("nombre")

'Instanciamos y abrimos nuestro objeto conexion
Set Conn = Server.CreateObject("ADODB.Connection")
Conn.Open "Mibase"

'Ahora creamos la sentencia SQL
sSQL="Delete From Clientes Where nombre='" & nombre & "'"

'Ejecutamos la orden
set RS = Conn.Execute(sSQL)
%>

<h1><div align="center">Registro Borrado</div></h1>
<div align="center"><a href="lectura.asp">Visualizar el contenido de la base</a></div>

<%
'Cerramos el sistema de conexion
Conn.Close
%>

</BODY>
</HTML>
```

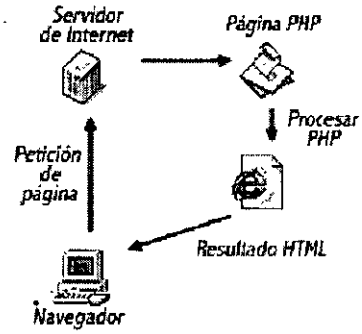
Hasta aquí trabajaremos con la tecnología de Microsoft, ahora bien, iniciemos los antecedentes de la programación con PHP, recuerde que, la finalidad es que usted identifique la tecnología que mas se adapte a su presupuesto y arquitectura tecnológica.

Programación con PHP



Coordinador Miguel Angel Flores
[illegible]@gmail.com

Instructor: Mauricio Angulo
mauricioangulo@[illegible]



Introducción:

PHP es un lenguaje interpretado diseñado para favorecer el desarrollo de web-stes dinámicos y aplicaciones para web-sites. La distribución más popular de PHP es como módulo para el web-server Apache, aunque puede funcionar con las limitaciones que ya conocemos, como un interprete para ejecutar aplicaciones Cgi en aquellos web-servers que no lo soporten como módulo. PHP se distribuye en formato open-source y es gratuito, una instalación habitual de PHP consiste en compilar el módulo PHP y luego recompilar el Apache para que utilice el módulo recientemente compilado.

Generalidades:

La característica más importante de PHP es que permite combinar código html y código php en una misma página (de extensión php), por ejemplo:

```
<HTML>
<HEAD><TITLE>Hola</TITLE></HEAD>
<BODY>
Hola esta es una prueba. <BR />
<?php
print("Hola soy una linea generada en php <BR />");
?>
</BODY>
</HTML>
```

Este ejemplo al guardarse en un archivo de extensión .php es automáticamente parseado por el interprete de php cuando el browser envía un pedido. El ciclo es el siguiente:

· El browser envía un pedido de un archivo con extensión php.

· El server analiza que la extensión del request es .php, obtiene el archivo y lo envía al interprete php.

· El interprete php del web-server parsea el archivo en busca de tags <? ?> y procesa todo lo que se encuentre entre dichos tags (puede haber varias apariciones de los tags en un mismo archivo), todo aquello que esta fuera de los tags se envía al browser sin interpretar.

· El resultado combinado de aquello que no debe interpretarse y el resultado del código interpretado se envía al browser.

En nuestro ejemplo el browser recibiría:

```
<HTML>
<HEAD><TITLE>Hola</TITLE></HEAD>
<BODY>
Hola esta es una prueba. <BR />
Hola soy una línea generada en php <BR />
</BODY>
</HTML>
```

Como podemos ver, es muy sencillo combinar código html y php. Para generar html desde php tenemos las siguientes opciones:

- **Usar la función "print de php"**

- **Usar la función "echo de php"**

- **Cerrar el tag ?> escribir el código html deseado y volver a abrir el tag <?**

Generación de web sites dinámicos usando PHP

La tercera opción es la más eficiente en velocidad cuando el código html que debemos generar es fijo, cuando el código html es dinámico podemos usar una mezcla de print y tags que abren y cierran que suele ser lo mas eficiente, por ejemplo:

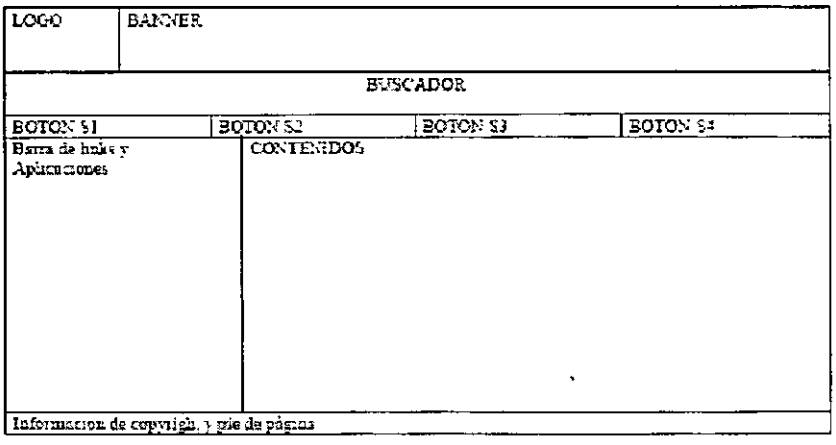
```
<form name="<? print("$nombre_form")?>">
etc…
```

Conceptos básicos en la generación de sites dinámicos con PHP.

Una característica interesante de php es que permite realizar "includes" dentro de un script php, de esta forma se puede modularizar una página o el layout de una página en varios módulos php que se desarrollan en forma independiente, además pueden desarrollarse en php componentes reusables que luego se utilizan en otras páginas o incluso en otros sites.

Una forma común de trabajo usando php para generar sitios dinámicos es definir "templates" o "layouts" en los cuales se divide la página en "zonas" o "módulos" que serán desarrollados en php, el layout de la página con la forma y tamaño de cada zona se puede definir sin problemas usando por ejemplo tablas de html.

A lo largo de este manual desarrollaremos a modo de ejemplo un mini-portal de noticias dinámico al cual le agregaremos servicios o aplicaciones a medida que se estudian distintas características de php. Supongamos que tenemos por el momento un único "template" o "layout" para nuestro sitio que determina la forma en la cual se vera la "home page" del mismo, el equipo de diseño nos entrega el siguiente layout:



Como podemos ver el site cuenta con 4 secciones que se acceden desde una barra navegadora ubicada debajo del search box, además existe una barra de links y servicios a la izquierda y una zona de contenidos que es la zona principal de la página.

Aun sin saber que funcionalidad tiene o de que forma se debe implementar cada parte podemos esquematizar el layout de la página usando php y html de la siguiente forma:

```
<HTML>
<HEAD>
<TITLE>Layout</TITLE>
Generación de web sites dinâmicos usando PHP
</HEAD>
<BODY>
<table width="640" border="1">
<tr><td width="20%">Logo</td><td colspan="3">Banner</td></tr>
<tr><td colspan="4">buscador</td></tr>
<tr><td>s1</td><td>s2</td><td>s3</td><td>s4</td></tr>
<tr><td width="20%">ColIzq</td><td colspan="3">Contenidos</td></tr>
</table>
</BODY>
</HTML>
Luego podemos reemplazar cada "zona" de la home page por un include
en php que generara
dinâmicamente la parte de la página en cuestión:
<HTML>
<HEAD>
<TITLE>Layout</TITLE>
</HEAD>
<BODY>
<table width="640" border="1">
<tr><td width="20%"><? Include("logo.php");?></td><td
colspan="3"><?include("banner.php");?></td></tr>
<tr><td colspan="4"><?include("buscador.php");?></td></tr>
<tr><?include("botonera.php");?></tr>
<tr><td width="20%"><?include("izq.php");?></td><td
colspan="3"><?include("contenidos_home.php");?></td></tr>
</table>
</BODY>
</HTML>
```

De esta forma hemos modularizado el layout de la página y tenemos como resultado que deben desarrollarse los siguientes módulos:

- **logo.php**
- **banner.php**
- **buscador.php**
- **botonera.php**
- **izq.php**
- **contenidos_home.php**

PHP es un lenguaje no posicional, por lo que no importa la columna en la cual se comience a escribir el código. Tampoco influye sobre el código la cantidad de saltos de línea (enter) que se coloquen, ni la cantidad de espacios. La forma en la que se separan las distintas sentencias es mediante la utilización de ";". En PHP cada sentencia debe finalizar con ";". Se puede escribir más de una sentencia en la misma línea siempre y cuando las mismas se encuentren separadas con ";". Comentarios:

En PHP hay 3 formas distintas de incluir comentarios:

```
/* Al estilo de C
```

en donde el comentario empieza
y termina delimitado por barra asterisco y asterisco barra

```
*/
```

O bien usando

```
// Comentario
```

O por último

```
# Comentario
```

En las dos últimas variantes el comentario empieza en donde se encuentra el "//" o el "#" y termina cuando termina la línea.

Tipos de Datos:
PHP soporta los siguientes tipos de datos:

- Enteros
- Vectores
- Binarios de punto flotante
- Strings
- Objetos

En general el tipo de dato de una variable no es decidido por el programador sino que lo decide el lenguaje en tiempo de ejecución, la instrucción settype puede usarse para forzar el tipo de dato de una variable en los raros casos en que esto sea necesario. Todas las variables en php se denotan utilizando el signo '$' precediendo al nombre de la variable.

Enteros:

```
$a = 1234; # número decimal
$a = -123; # número negativo
$a = 0123; # número octal (83 en decimal)
$a = 0x12; #número en hexadecimal (18 decimal)
```

Flotantes:
Los números de punto flotante pueden notarse de la siguiente manera:

```
$a = 1.234;
$a = 1.2e3;
```

Strings:
En PHP los strings tienen un manejo similar al utilizado en "C" o "C++", están predefinidos los siguientes caracteres especiales:

```
\n Nueva línea
\r Salto de carro (carring return)
\t Tabulación
\\ Barra invertida
\$ Signo pesos
\" Comillas doble
```

Un string puede inicializarse usando comillas dobles o comillas simples. Cuando se utilizan comillas dobles el interprete de php parsea previamente el string, es decir que reemplaza los nombres de variables que encuentra en el string por el contenido de la variable. Cuando se usan comillas simples el string se imprime tal y como figura sin ser parseado.
Ejemplo:

```
$x="Juan";
$s="Hola $x";
$t='Hola $x'
$s vale "Hola Juan" y $t vale "Hola $x".
```

Otra forma de inicializar un string es usando un string multilinea de la siguiente manera:

```
$str=<<<EOD
```

Este es un ejemplo de un string que ocupa varias líneas y se puede definir así
EOD; Se pueden concatenar strings usando el operador "." de la siguiente manera:

```
$x="hola";
$y="mundo";
$s=$x.$y; #$s es igual a "holamundo".
$s=$x." ".$y; # Aquí $s vale "hola mundo"
```

Vectores:

Los vectores en php actúan tanto como vectores tradicionales (indexados por número) así también como
vectores asociativos (indexados por clave).
Los vectores pueden crearse usando las instrucciones "list" o "array" o bien inicializando en forma explícita
cada elemento del vector.

```
$a[0]="hola"
$a[1]="mundo;
$a["clave"]="valor";
```

Utilizando la notación especial $v[]; se pueden agregar elementos a un vector.

```
$a[0]="nada";
$a[1]="hola";
$a[]="mundo"; #Asigna a $a[2] el valor "mundo".
```

Existen funciones especiales para ordenar vectores, contar la cantidad de elementos de los mismos, recorrerlos, etc. (Ver el capítulo sobre vectores)

Matrices:

La definición, inicialización y uso de matrices en PHP es sencilla. Se puede pensar una matriz en PHP como un vector de vectores, por lo que se puede utilizar la misma lógica que en los primeros.

```
$a[0][1]="Hola";
$a[0]["clave"]="una cosa";
$a["clave1"]["clave2"][0][1]="otra cosa";
```

etc...
Para incluir el valor de un elemento de un vector en un string se deben usar llaves para indicar el alcance del nombre de la variable a reemplazar:

```
Echo "Esta es una prueba {$a[0][1]}";
```

Una forma útil de inicializar un vector asociativo es usando la siguiente notación:

```
$a=array(
"color" => "rojo",
"edad" => 23,
"nombre" => "juan"
);
```

Para crear una matriz se pueden anidar las declaraciones de tipo array.

```
$a = array(
"apple" => array(
"color" => "red",
"taste" => "sweet",
"shape" => "round"
),
"orange" => array(
"color" => "orange",
"taste" => "tart",
"shape" => "round"
),
"banana" => array(
"color" => "yellow",
"taste" => "paste-y",
"shape" => "banana-shaped"
)
);
```

Constantes:

Para definir una constante se utiliza la instrucción "define" de la forma:

```
define("PI",3.14151692);
```

Luego las constantes pueden usarse como variables tradicionales ($PI) con la salvedad de que no se les puede asignar un valor.

Operadores:

Los operadores aritméticos en PHP también se asemejan al C:

```
$a + $b; //suma
$a - $b; //resta
$a++; //pos-incremento, esta sentencia devuelve el valor de $a y lo
incrementa en 1
++$a; //pre-incremento, incrementa en 1 el valor de $a y devuelve
el valor incrementado.
$a--; //pos-decremento
--$a; //pre-decremento
$a * $b; //multiplicación
$a / $b; //división
$a % $b; //módulo
```

Asignación:

La asignación se resuelve con el signo igual ("=").

```
$a=5; //Asigna 5 a $a
$a=$b; //Asigna el valor de $b a $a
$b=($c=6); //Asigna 6 a $c y a $b
```

Y pueden combinarse asignación y operadores de la forma:

```
$a+=5; //Suma y asigna 5 a $a
$x.="hola"; //Concatena $x con "hola"
Operaciones con bits:
$a & $b; //$a AND $b
$a | $b; //$a OR $b
$a ^ $b; //$a XOR $b
~$a; //NOT $a
$a << $b; //Shift hacia la izquierda $b posiciones
$a >> $b; //Shift hacia la derecha $b posiciones
Comparaciones:
$a = = $b; //true si $a igual a $b
$a = = = $b; //true si $a igual a $b y además son del mismo tipo
$a >= $b; //mayor o igual
$a <= $b; //menor o igual
$a != $b; //true si a y b son distintos Operador @
```

Cuando se antepone @ a una expresión se suprimen los errores que la expresión pudiera generar.
Ejemplo:

```
@($c=$a/$b);
```

Operador de ejecución:

PHP soporta el uso de "backticks" (comillas invertidas) para ejecutar un comando desde el shell y devolver el resultado de la ejecución del comando en una variable:

```
$result=`ls -l`;
```

Operadores lógicos:

```
$a && $b; //True si $a es true y $b es true
$a || $b; //True si $a es true o $b es true
$a xor $b; //Or exclusivo
!$a; //True si $a es falso (NOT)
```

Estructuras de Control:

```
If:
if (expresión) sentencia;
if (expresión) {sentencias;}
if (expresión) {
sentencias;
} else {
sentencias;
}
if (expresión) {
sentencias;
} elseif (expresión) {
sentencias;
} else (expresión) {
sentencias;
}
While:
while (expresión) {
sentencias;
}
do {
sentencias;
} while(expresión)
For:
for (expr1,expr2,expr3) {
sentencias;
}
```

La primera expresión cumple la función de inicializar las variables de control del FOR. Esta expresión se cumple incondicionalmente, más allá de que se entre dentro del ciclo o no.
La expresión 2 se evalúa siempre que se este por ingresar al ciclo del FOR, aún cuando se ingresa al for por primera vez.
La tercera expresión se ejecuta cada vez que se termina el ciclo. Por lo general se utiliza esta expresión para indicar el incremento de alguna variable que se este utilizando para el FOR. La ejecución de esta expresión es también incondicional y es la que se ejecuta inmediatamente antes de evaluarse la expresión 2.

Ejemplo:

```
for ($i=0;$i<5;$i++) {
print("$i");
}
```

Foreach:
Para realizar ciclos con la cantidad de ocurrencias en un vector se utiliza el comando foreach:

```
foreach ($vector as $variable) {
sentencias;
}
foreach ($vector as $clave => $valor) {
sentencias;
}
Por cada iteración cada elemento de $vector es asignado a
$variable.
El segundo caso es aplicable para recorrer vectores asociativos.
Break:
Break permite salir del ciclo actual "for" , "while" o "switch"
Ejemplo:
for ($i=0;$i<6;$i++) {
If($i==$b) break; //Sale del ciclo si $i es igual a $b
}
Switch:
El switch permite ejecutar un grupo de sentencias de acuerdo al
valor de una variable:
switch ($variable) {
case valor:
sentencias;
break;
case valor2:
sentencias;
break;
default:
sentencias;
break;
}
```

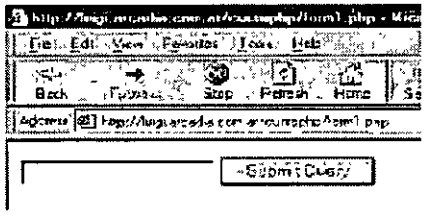
Cuando el valor de la variable ($variable) coincide con el valor de algún "case", se ejecutan las sentencias que se encuentran a continuación. En este caso se utiliza la sentencia "break" en forma prácticamente obligatoria, porque en caso de no existir esta sentencia se seguiría ejecutando linealmente todas las sentencias continuas, es decir las sentencias de los demás "cases" inferiores. Por último, la opción "default" se utiliza generalmente para cuando el valor de la variable no coincide con ningún "case". Estas sentencias se ejecutan siempre, salvo en el caso de que se ejecute antes un "break".

Manejo de Forms.
El mecanismo básico de interacción entre el usuario y un web-site esta dado por el uso de formularios html, el server envía un formulario que el browser muestra en pantalla permitiendo al usuario ingresar datos, luego los datos en el formulario viajan al server en el próximo request realizado por el browser para ser procesados en el mismo. La respuesta del server suele depender de los datos recibidos en el formulario.

El siguiente es un ejemplo de formulario en HTML usando un campo de entrada de tipo "text"

```
<FORM ACTION="procesar.php" METHOD="POST">
<INPUT TYPE="text" NAME="texto">
<INPUT TYPE="submit" NAME="proc">
</FORM>
```

Este formulario HTML se ve en pantalla de la siguiente manera:



Una vez que el usuario ingresa un texto y presiona el botón de submit el browser genera un request con método "Post" al script "procesar.php" que es el script que se va a encargar de procesar los datos ingresados en el formulario.
Dentro del script php los datos del formulario se reciben en variables php que tienen el mismo nombre que los indicados con "NAME" en el formulario, en este caso el script recibe $texto con el texto tipeado por el usuario en el formulario.

El script que recibe el formulario podría por ejemplo ser:

```
(procesar.php)
<?
print ("El valor ingresado en el formulario es: $texto <BR />");
?>

En PHP es posible que un form se procese a si mismo, esto lo
podemos hacer de la siguiente manera:

(form1.php)
<?
if(isset($proc)) {
print("el valor ingresado es: $texto");
} else {
?>
<FORM ACTION="form1.php" METHOD="POST">
<INPUT TYPE="text" NAME="texto">
<INPUT TYPE="submit" NAME="proc">
</FORM>
<?
} //Esto cierra el else que abrimos arriba.
?>
```

Notar que el nombre del script que muestra el formulario es el mismo que el script usado en "action" para procesarlo, la instrucción isset de PHP devuelve true si la variable esta seteada, para un formulario si el usuario presiona el botón de submit se setea automáticamente la variable que corresponde al "NAME" del botón submit del formulario, por eso preguntamos si esta seteado $proc para saber si hay que mostrar el formulario o procesarlo. Podría también procesarse el formulario y a su vez mostrarlo o mostrar otro distinto, las variantes dependen de que es lo que se quiere hacer.

Text type:

Para ingresar texto mediante un formulario html se usa el tag input con atrobuto type="text", los atributos disponibles son:

Atributos:

```
maxlength="numero" Cantidad máxima de caracteres que se pueden
ingresar
name="text" Nombre de la variable php que recibirá el valor
size="numero" Tamaño del campo de entrada a mostrar en pantalla
value="texto" Valor inicial a mostrar en el campo de entrada
(default)
```

Hidden Type:

El tag input con type="hidden" funciona en forma idéntica a un tipo "text" con la salvedad de que no se muestra en pantalla, esto es útil para pasar variables entre formularios o guardar variables "ocultas" en un formulario.

Checkboxes:
Los checkboxes son campos de entrada que soportan solamente los estados de seteado o no. Para ello se usa el tag input con type="checkbox" , los atributos disponibles son:

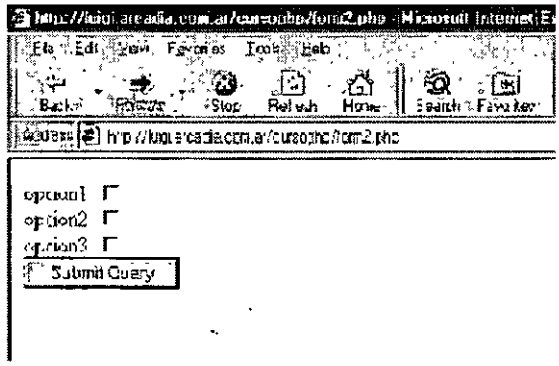
Atributos:

Checked Si el atributo esta presente el checkbox aparecerá marcado por default.
name="text" Nombre de la variable php que recibe el valor.
value="text" Valor que toma la variable si esta seteada, el default es "on"
El script que recibe los resultados sólo recibe los nombres de los checkboxes que están seteados, es común en php generar una lista de checkboxes a partir de un vector, veamos un ejemplo:

```
(form2.php)
<FORM ACTION="form2.php" METHOD="post">
<?
$vector=array("opcion1","opcion2","opcion3");
for ($i=0;$i<count($vector);$i++) {
print("$vector[$i]");
?>
<input type="hidden" name="valor[<?print($i);?>]"
value="<?print("$vector[$i]");?>">
<input type="checkbox" name="vector[<?print($i);?>]"> <br>
<?
}
?>
<INPUT TYPE="submit" name="proc">
</FORM>
```

Este es un ejemplo muy útil en el cual el formulario html no es siempre el mismo sino que es generado dinámicamente desde php en base a por ejemplo el contenido de un vector.

El formulario se muestra en el browser de la siguiente manera:



Y el código html que recibe el browser para mostrar el formulario
(que se genera en el servidor) es:

```
<FORM ACTION="form2.php" METHOD="post">
opcion1 <input type="hidden" name="valor[0]" value="opcion1">
<input type="checkbox" name="vector[0]"> <br>
opcion2 <input type="hidden" name="valor[1]" value="opcion2">
<input type="checkbox" name="vector[1]"> <br>
opcion3 <input type="hidden" name="valor[2]" value="opcion3">
<input type="checkbox" name="vector[2]"> <br>
<INPUT TYPE="submit" name="proc">
</FORM>
```

Observar el uso de campos de texto ocultos para indicar cual es el valor de un textbox en caso de estar seleccionado, también podríamos haber usado el campo value de los checkboxes, es otra forma de hacer lo mismo. Le podemos agregar al formulario la opción de mostrar cuales son los checkboxes seleccionados usando:

(Agregar este código al principio de **form2.php**)

```
<?
if(isset($proc)) {
for ($i=0;$i<count($valor);$i++) {
if(isset($vector[$i])) {
if($vector[$i]=="on") {
print("$valor[$i] viene seleccionado");
}
}
}
}
?>
```

Como resultado el script informa cuales son los checkboxes que han sido seleccionados por el usuario y cuales no.

El script completo es:

(form2.php)

```
<?
if(isset($proc)) {
for ($i=0;$i<count($valor);$i++) {
if(isset($vector[$i])) {
if($vector[$i]=="on") {
print("$valor[$i] viene seleccionado");
}
}
}
}
?>
<FORM ACTION="form2.php" METHOD="post">
<?
$vector=array("opcion1","opcion2","opcion3");
for ($i=0;$i<count($vector);$i++) {
print("$vector[$i]");
?>
<input type="hidden" name="valor[<?print($i);?>]"
value="<?print("$vector[$i]")
;?>">
<input type="checkbox" name="vector[<?print($i);?>]"> <br>
<?
}
```

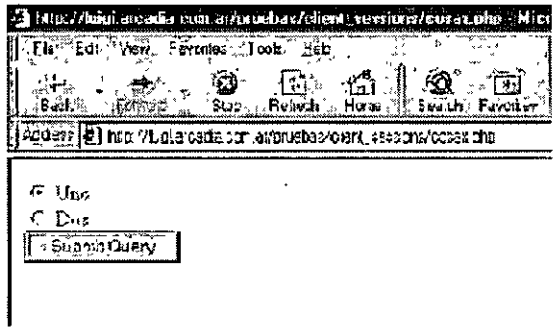
Radio Buttons:

Un radio button también es un tipo de botón de 2 estados (encendido-apagado) pero estos pueden agruparse de forma tal que sólo un radio-button del grupo pueda estar prendido. El nombre de radio button fue puesto por su funcionamiento parecido a los botones de las viejas radios de automóviles, que siempre debía estar presionado uno de los botones y nunca (mientras funcionaba correctamente) podían estar dos presionados al mismo tiempo.

Los atributos son checked, name y value. Todos los radio buttons del mismo nombre pertenecen al mismo grupo, es decir que de todo el grupo sólo uno podrá ser seleccionado. El value del botón determina cual es el radio elegido del grupo. Ejemplo:

```
<?
if(isset($proc)) {
print("El radio seteado es $grupo <BR />");
}
?>
<FORM ACTION="form3.php" METHOD="post">
<input type="radio" name="grupo" value="uno" checked> Uno <br>
<input type="radio" name="grupo" value="dos"> Dos <br>
<INPUT TYPE="submit" name="proc">
</FORM>
```

El formulario que se despliega en pantalla es:



Y solamente uno de los dos radio buttons puede estar habilitado.

Image:

Es posible reemplazar el boton submit por una imagen gif o jpg de forma de darle al botón el look and feel que se quiera, esto se hace con el tag input type="image" los atributos son:

```
name="texto" Cumple la misma función que el atributo name de
submit.
src="url" URL de la imagen a mostrar, ejemplo: "images/boton.jpg"
```

TextArea:

Un textarea permite ingresar texto en una caja de formato mayor a un input type="text" , la notación de un textarea es distinta a la de un tag de input. Los atributos son los siguientes:

```
cols="numero"      Número de columnas del textarea
rows="numero"      Número de líneas
name="texto"       Nombre de la variable que recibe el texto
wrap="off/virtual/physical"  Forma en la cual se cortan las
palabras cuando termina cada línea, off elimina el corte de
palabras, virtual muestra los cortes pero estos no son transmitidos
al server, physical cortalas palabras y además transmite los saltos
de línea al server.
```

Ejemplo:

```
<textarea name="texto" cols="20" rows="10">
</textarea>
```

Entre el tag que abre y cierra si se desea se puede poner texto que se muestra en el textarea y el usuario puede modificar.

File Uploads

El último tipo de dato que se puede transferir al server usando un formulario es un archivo, este es un tipo de transferencia especial pues implica generar un archivo en el file-system del web-server a partir de un archivo que el usuario selecciona desde su disco local.

HTML soporta el upload de archivos usando el tag <input> con type="file", este tipo de input genera un boton "browse" en el browser que permite al usuario seleccionar un archivo desde su file-system local (usando una caja de navegación por los discos standard del sistema operativo).

El formulario para subir un archivo es de la forma:

```
<FORM ENCTYPE="multipart/form-data" ACTION="upload.php"
METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

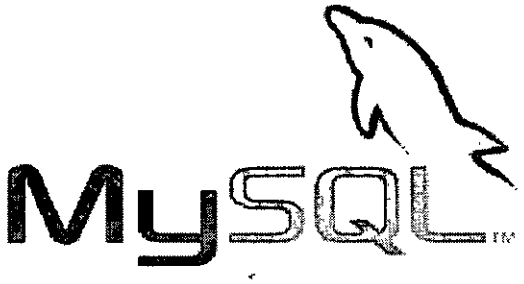
Como puede verse hay un campo oculto que indica cual es el limite máximo de tamaño que se puede subir, este valor es chequeado en el "cliente", además PHP dispone de una variable que se inicializa en el archivo de configuración de php (en gral /var/lib/php.ini) allí se limita el tamaño máximo de los uploads al llegar al "server"

El script **upload.php** que recibe los datos del formulario recibe las siguientes variables:

- $userfile - Path del archivo almacenado en el server.
- $userfile_name - Nombre del archivo segun el usuario
- $userfile_size - Tamaño del archivo subido
- $userfile_type - Mime type del archivo, por ejemplo image/gif

El script que recibe el archivo es responsable de hacer lo que corresponda con el mismo ya que en general el archivo se almacena en un directorio temporal y es eliminado una vez que termina el script. El script debe almacenar el archivo en una base de datos, moverlo a un directorio permanente, tomar datos de el o realizar el procesamiento que corresponda.

Manejo de Bases de Datos -MySQL-



Coordinador Miguel Angel Flores
miguelangelflores@gmail.com

Instructor: Mauricio Angulo
mauricioangulo@terra.com.mx

Antecedentes

Una de las características más importantes de PHP es su integración con diversos motores de base de datos. El PHP está construido para generar en forma sencilla páginas web dinámicas a partir de información almacenada en bases de datos. A continuación mostramos las funciones más importantes y ejemplos típicos de uso para una base MySql.

1. Conexión a la base

```
$db_link=mysql_connect(hostname, user, password);
```

Ejemplo:

```
$db=mysql_connect("localhost","root","secreto");
```

La función realiza la conexión a la base Mysql y devuelve "false" si hubo algún error en la conexión o un link a la conexión a la base en caso de que la conexión sea exitosa.

El link es un número que indica la sesión dentro del MySql. Esta sesión se mantiene hasta que [...]. Para finalizar la conexión se debe utilizar la función mysql_close().

Es muy importante cerrar la conexión a la base de datos una vez finalizadas las transacciones para evitar la sobrecarga en el motor de la base de datos.

2. Selección de la base de datos a utilizar

```
mysql_select_db(database_name, db_link);
```

Ejemplo:

```
mysql_select_db("test",$db);
```

Esta función configura cual es la base de datos que se utilizara por omisión. En este caso el link a utilizar en esta función es el link que se obtuvo al ejecutar la función mysql_connect.
La función mysql_select_db devuelve el valor "false" en caso de que se encuentre algún error, como por ejemplo la inexistencia de la base de datos.

En este punto cabe aclarar que la denominación de las bases de datos de MySql es case-sensitive, por lo que debemos mantener un standard a la hora de elegir los nombres de las distintas bases de datos.

3. Queries a la base de datos.

```
$result=mysql_query(query,db_link);
```

Ejemplo:

```
$result=mysql_query("update clientes set deudor='si' where
apellido='Picapiedras', $db)
$query="insert into clientes (nombre, Apellido) values (Pedro,
Mármol) ";
$result=mysql_query($query,$db);
```

Nuevamente el link que se debe usar es el que se obtiene al conectarse a la base, mysql_query devuelve falso en caso de que el query no pueda ejecutarse (error de SQL) o bien un result set en los casos que devuelva algún tipo de datos como por ejemplo en un select.

Es muy importante fijarse con que usuario se realizó la conexión a la base de datos a la hora de ejecutar el mysql_connect, ya que la gran mayoría de los errores producidos en esta instancia son el resultado de la falta de permisos para realizar la consulta.

4. Cantidad de Filas Consultadas o Modificadas

4.1 Filas Consultadas

```
$cantidad=mysql_num_rows($result);
```

Ejemplo:

```
$query="select nombre, telefono from contactos where edad between
20 and 25 and sexo='f'";
$result=mysql_query($query,$db);
$cant=mysql_num_rows($result);
```

Esta función devuelve la cantidad de filas que se obtuvieron luego de ejecutar una instrucción de consulta como por ejemplo la función select.

En el caso del ejemplo, en la variable $cant nos dirá cuantas chicas de entre 20 y 25 años tenemos en nuestra lista de contactos.

4.2 Filas Modificadas

```
$cantidad=mysql_affected_rows(db_link);
```

Ejemplo:

```
$cuantos=mysql_affected_rows($db);
```

Devuelve cuantos registros fueron afectados por un query con insert, update, o delete, notar que se le pasa el db_link ya que el result_set no tiene sentido. Si el query fue un delete sin clausula "where" esta función devuelve cero independientemente del número de registros eliminados de la tabla.

5. Obtención de registros de una consulta

5.1 Obtención de datos en un result set

```
$var=mysql_fetch_row(result_set);
```

Ejemplo:

```
$query="select nombre, telefono from contactos where edad between
20 and 25 and sexo='f'";
$result=mysql_query($query,$db);$rs=mysql_fetch_row($result);
```

Toma un registro del result set y lo devuelve en un vector en el cual el elemento con índice 0 es la primer columna del registro, el elemento con índice 1 es la segunda columna, etc. Si no hay más registros por devolver devuelve false.

Los valores de los campos solicitados en el result set son devueltos en forma de array, es decir que para acceder al valor de nombre de la primera fila de nuestro ejemplo debo llamar a la variable $rs[0].

Ejemplo:

```
while($v=mysql_fetch_row($result)) {
print("Columna 0: $v[0] Columna 1: $v[1]");
}
```

5.2 Obtención de datos en un vector

```
$query="select nombre, telefono from contactos where edad between
20 and 25 and sexo='f'";
$result=mysql_query($query,$db);
$rs=mysql_fetch_array($result);
Ejemplo:
$rs=mysql_fetch_array($result);
```

Funciona de forma idéntica a mysql_fetch_row pero devuelve los resultados en un vector asociativo indexado por nombre de columna, por lo que para obtener los nombres de nuestra consulta es necesario llamar al vector de la forma $rs["nombre"]

Ejemplo:

```
while($v=mysql_fetch_array($result)) {
print("Columna 0: $v["nombre"]");
}
```

6. Cerrar la conexión

```
mysql_close(db_link)
```

Por último, esta función nos sirve para cerrar la conexión a la base.

Tablas para otras funciones

Función	Descripción
result=mysql_create_db(database_name,db_link),	Crea una base de datos con el nombre indicado, devuelve true o false según el resultado.
Result=mysql_data_seek(result_set, position).	Mueve el puntero interno de cada result set a la posición indicada (el primer registro es la posición 0), de esta forma la próxima llamada a mysql_fetch_row o mysql_fetch_array devolvera el registro que se acaba de apuntar. Devuelve true o false.
Result=mysql_db_query(database_name, query, db_link).	Realiza una consulta a una base indicada, es obviamente reemplazable por un mysql_select_db y un mysql_query aunque resulta útil si hay que consultar tablas en varias bases distintas.
Result=mysql_drop_db(database_name,db_link);	Dropea la base con el nombre indicado, devuelve true o false
Object=mysql_fetch_field(result_set, numero_columna);	Devuelve un objeto con información sobre la columna indicada de un result set (0 es la primera columna, 1 la segunda, etc). Las propiedades que se setean en el objeto son: • name – nombre de la columna • table – nombre de la tabla • max_length – longitud maxima de la columna • not_null – 1 si la columna no puede ser null • primary_key - 1 si la columna es primary key • unique_key - 1 si la columna es unique key • multiple_key - 1 si la columna no es unique key • numeric - 1 si la columna es numerica • blob - 1 si la columna es un BLOB • type – Tipo de la columna • unsigned - 1 si la columna es sin signo • zerofill - 1 si la columna es zero-filled
object=mysql_fetch_object(result_set);	Similar a mysql_fetch_array con la diferencia de que los resultados se devuelven en un vector en lugar de un vector asociativo. Luego se puede acceder a las distintas columnas del registro devuelto como propiedades (data_members) del objeto devuelto.
int=mysql_num_fields(result_set),	Devuelve el numero de columnas en un result set

Mensaje de errores

```
Mensaje=mysql_error();
```

Devuelve un texto con el mensaje correspondiente al error que ocurrió en caso de que se produzca un error en la base de datos.

Ejemplo:

```
$result=mysql_query($query,$db);
if(!$result) {
$x=mysql_error();
die(" Ocurrio un error en la base de datos: $x");
}
```

5. Generación de una pagina a partir de una base de datos

Repasemos el layout de nuestro web-site ejemplo:

<table>
<tr><td>LOGO</td><td colspan="4">BANNER</td></tr>
<tr><td colspan="5">BUSCADOR</td></tr>
<tr><td>BOTON S1</td><td colspan="2">BOTON S2</td><td>BOTON S3</td><td>BOTON S4</td></tr>
<tr><td>Barra de links y Aplicaciones</td><td colspan="4">CONTENIDOS</td></tr>
<tr><td colspan="5">Informacion de copyrigh, y pie de página</td></tr>
</table>

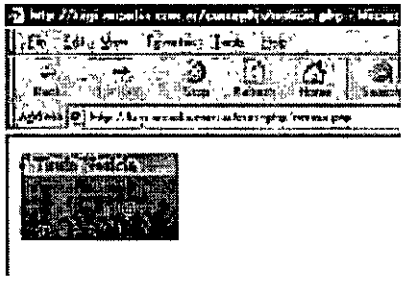
Supongamos que a la barra de links y aplicaciones a la derecha se decide agregarle un módulo "noticia de la semana" el layout queda entonces de la siguiente manera:

<table>
<tr><td>LOGO</td><td colspan="4">BANNER</td></tr>
<tr><td colspan="5">BUSCADOR</td></tr>
<tr><td>BOTON S1</td><td colspan="2">BOTON S2</td><td>BOTON S3</td><td>BOTON S4</td></tr>
<tr><td colspan="2">Noticia de la semana.</td><td colspan="3" rowspan="2">CONTENIDOS</td></tr>
<tr><td colspan="2">Barra de links y aplicaciones.</td></tr>
<tr><td colspan="5">Información de copyrigh, y pie de página</td></tr>
</table>

Se define que el módulo php que mostrará la noticia de la semana se denominará noticia.php

Ejercicio: Modificar el archivo **layout1.php** para reflejar el cambio.

A continuación una tarea habitual es que el equipo de diseño realice un archivo html (o php) con un ejemplo de cómo debe quedar para que sea visualmente agradable el módulo noticia de la semana, por ejemplo puede verse de la siguiente manera:



El código HTML del módulo **noticia.php** es el siguiente:

```
<table bgcolor="#22AE22" width="140" border="0" cellspacing="0"
cellpadding="0">
<tr><td>
<img src="flechita.gif">
<b>Título Noticia</b>
</td></tr>
<tr><td bgcolor="#226622">
Información sobre la noticia datos varios cosas divertidas.
</td>
</tr>
</table>
```

Lo que debemos hacer es reemplazar el título de la noticia y el texto de la misma por valores que obtenemos de la base de datos. Para ello decidimos crear una tabla llamada "noticia" en donde almacenaremos la noticia de la semana, además tenemos que construir un cargador que permita insertar el título y texto de la noticia en la base.

Definimos la tabla noticia en MySQL de la siguiente manera:

```
CREATE TABLE noticia(
título varchar(40),
texto text,
fecha datetime
);
```

La fecha la vamos a usar para poder insertar varias noticias y que en la página se muestre siempre la última (la que tiene fecha más reciente), el historial de noticias de la semana puede usarse eventualmente en otra página para mostrar un listado de los títulos de la noticia de la semana, etc. El formulario cargador que vamos a utilizar usa las técnicas que vimos en el capítulo de formularios y le agregamos el manejo de la base de datos para que realice el insert en la tabla de noticias.

```
<?
//Este código se ejecuta cuando se presiono el botón de submit
if(isset($proc)) {
$dab=mysql_connect(" localhost","root","seldon");
mysql_select_db("curso",$dab);
$query="INSERT into noticia(título,texto,fecha)
values('$título','$texto',now()
)";
$res=mysql_query($query,$dab);
if(!$res) {
$x=mysql_error();
print("Se produjo un error al insertar: $x");
}
}
?>
<form action="<?print("$PHP_SELF");?>" method="post">
Título de la noticia: <input type="text" maxlength="40"
name="título"> <BR>
Texto de la noticia: <textarea rows="10" cols="80"
name="texto"></textarea><BR>
<input type="submit" name="proc">
</form>
```

Luego podemos modificar el módulo **noticia.php** para que obtenga el título y texto de la última noticia y sea esto lo que se muestre al usar el módulo.

Noticia.php modificado queda de la forma:

```
<?
//Obtenemos los datos de la noticia a buscar
$dab=mysql_connect("localhost","root","seldon");
if(!$dab) {$título="No hay noticia";$texto="No hay noticia";} else
{
mysql_select_db("curso",$dab);
$query="select título,texto from noticia order by fecha desc";
$res=mysql_query($query,$dab);
if(!$res) {$título="No hay noticia";$texto="No hay texto";} else {
$r=mysql_fetch_row($res);
$título=$r[0];
$texto=$r[1];
}
}
?>
<table bgcolor="#22AE22" width="140" border="0" cellspacing="0"
cellpadding="0">
<tr><td>
<img src="flechita.gif">
<b><?print("$título");?></b>
</td></tr>
<tr><td bgcolor="#226622">
<?print("$texto");?>
</td>
</tr>
</table>
```

Como podemos ver probando el módulo o usando el layout modificado con la inclusión de este módulo a partir de este momento la noticia de la semana que se muestra al ingresar a la página es la última noticia ingresada en la base, usando el cargador de noticias puede modificarse la misma cargando una nueva noticia y la próxima vez que se acceda a la base automáticamente se ve la nueva noticia. Este es el principio básico de un web-site dinámico y de la generación dinámica y en tiempo real de páginas web.

Almacenamiento de objetos binarios en MySQL

Un tema interesante en Mysql es usar la base de datos para guardar además de datos de tipo texto datos binarios como por ejemplo código html o imágenes. De esta forma se pueden almacenar completamente en la base de datos todos los elementos necesarios para construir un web-site.

```
CREATE TABLE binary_data (
id INT(4) NOT NULL AUTO_INCREMENT PRIMARY KEY,
description CHAR(50),
bin_data LONGBLOB,
filename CHAR(50),
filesize CHAR(50),
filetype CHAR(50)
);
```

El siguiente script puede usarse para insertar objetos binarios en la base de datos desde un browser. Notar que se usa el tag input type="file" de un form html para subir un archivo.

```
<HTML>
<HEAD><TITLE>Store binary data into SQL Database</TITLE></HEAD>
<BODY>
<?php
if ($submit) {
//codigo que se ejecuta si se presiono el botón submit
MYSQL_CONNECT( "localhost", "root", "password");
mysql_select_db( "binary_data");
$data = addslashes(fread(fopen($form_data, "r"),
filesize($form_data)));
$result=MYSQL_QUERY( "INSERT INTO binary_data
(description,bin_data,filename,filesize,filetype) ".
"VALUES
('$form_description','$data','$form_data_name','$form_data_size','$
form_d
ata_type')");
$id= mysql_insert_id();
print "<p>Database ID: <b>$id</b>";
MYSQL_CLOSE();
} else {
// sino mostrar el formulario para nuevos datos:
?>
<form method="post" action=" <?php echo $PHP_SELF; ?>"
enctype="multipart/form-data">
File Description:<br>
<input type="text" name="form_description" size="40">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000000">
<br>File to upload/store in database:<br>
<input type="file" name="form_data" size="40">
<p><input type="submit" name="submit" value="submit">
</form>
<?php
}
?>
</BODY>
</HTML>
```

Notar el uso de la variable predefinida $PHP_SELF que tiene el nombre del script de esta forma el formulario se llama a si mismo independientemente del nombre que se le ponga al archivo.
El siguiente script (getdata.php) puede usarse para recuperar los datos desde la base de datos, notar que el script espera recibir la variable $id con el id del registro a recuperar de la tabla.

```
<?php
if($id) {
@MYSQL_CONNECT( "localhost", "root", "password");
@mysql_select_db( "binary_data");
$query = "select bin_data,filetype from binary_data where id=$id";
$result = @MYSQL_QUERY($query);
$data = @MYSQL_RESULT($result,0, "bin_data");
$type = @MYSQL_RESULT($result,0, "filetype");
Header( "Content-type: $type");
echo $data;
};
?>
```

Para usar una imagen que se obtiene de la base de datos se puede usar:

```
<img src="getdata.php?id=3">
```

Notar como se le pasa la variable id al script para saber cual es el registro a recuperar de la base.

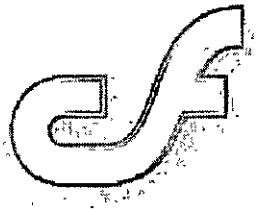
Usando este conjunto de scripts es sencillo:

- Armar un cargador que permita insertar en la base de datos archivos binarios (imágenes, código html, etc)
- Utilizar los datos almacenados en la base para insertarlos en una página web usando el script getdata o una variante del mismo.

Otras funcionalidades de PHP:

- Funciones de calendario y manipulación de calendarios usando MCAL
- Programación orientada a objetos
- Funciones para creación de archivos PDF
- Funciones de manejo de cajeros cybercash
- Parser de documentos XML
- WDDX
- Funciones de compresión de datos
- Manejo de archivos DBM
- Funciones para manipulación de fechas
- Funciones para manejo de directorios
- Funciones de encripción de datos
- Funciones de acceso al filesystem
- Funciones para manejo de FTP
- Funciones de hashing
- Generación dinámica de imágenes
- Manejo de cuentas de mail IMAP y POP3
- Funciones para envió de mail
- Funciones de networking usando sockets
- Funciones matemáticas
- Serialización de estructuras de datos
- Acceso a bases de datos (Mysql, Oracle, Postgress, Sybase, etc)
- Manejo de expresiones regulares.
- Manejo de sesiones.

Instalación del servidor



macromedia
COLDFUSION

Coordinador: Miguel Angel Flores
miguelfloresg@gmail.com

Instructor: Mauricio Angulo
mauricio@desarrollaweb.com

ColdFusion, combina un lenguaje intuitivo, basado en tags, rico, con herramientas visuales y un servidor de aplicaciones web probadamente confiable, para entregar la manera más rápida de desarrollar poderosas aplicaciones web.

ColdFusion es una herramienta que corre en forma concurrente con la mayoría de los servidores web de Windows, Linux y Solaris (también en servidores web personales en Windows 98 y puede ser usado para intranets). El servidor de aplicaciones web de ColdFusion trabaja con el servidor HTTP para procesar peticiones de páginas web. Cada vez que se solicita una página de ColdFusion, el servidor de aplicaciones ColdFusion ejecuta el script o programa contenido en la página.

ColdFusion es un lenguaje de programación, puede crear y modificar variables igual que en otros lenguajes de programación que nos son familiares. Posee controles de flujo de programas, como IF, Switch Case, Loop,etc. Tiene muchas funciones built-in para realizar tareas más complicadas como averiguar que día caerá el **3 de Agosto del 2007**

```
"DayOfWeekAsString(DayOfWeek('2007/08/03'))".
```

No es un lenguaje de bases de datos, pero interactúa de manera simple con bases de datos (Sybase, Oracle, MySQL,SQL, o Access). Usando SQL estándar, las páginas y aplicaciones web pueden fácilmente recuperar, guardar, formatear y presentar información dinámicamente.

ColdFusion es un lenguaje basado en tags, si te sientes cómodo con HTML, te encantará CFML (ColdFusion Markup Language). Muchas de las funciones poderosas de ColdFusion, como leer desde y escribir en discos duros del servidor, son basadas en tags.

Así como el tag <Table> puede tener argumentos como 'width' o 'align', el tag <CFFILE> tiene argumentos que especifican 'action=read/write/copy/delete', path=' etc.

ColdFusion integra tecnologías. ¿No sería agradable si no tuvieras que escribir todo el JavaScript para tus páginas?. El tag <CFFORM> construirá automáticamente todo el código JavaScript para verificar los campos requeridos antes de hacer submit al form. ColdFusion también tiene tags para COM, Corba y Applets y Servlets de Java.

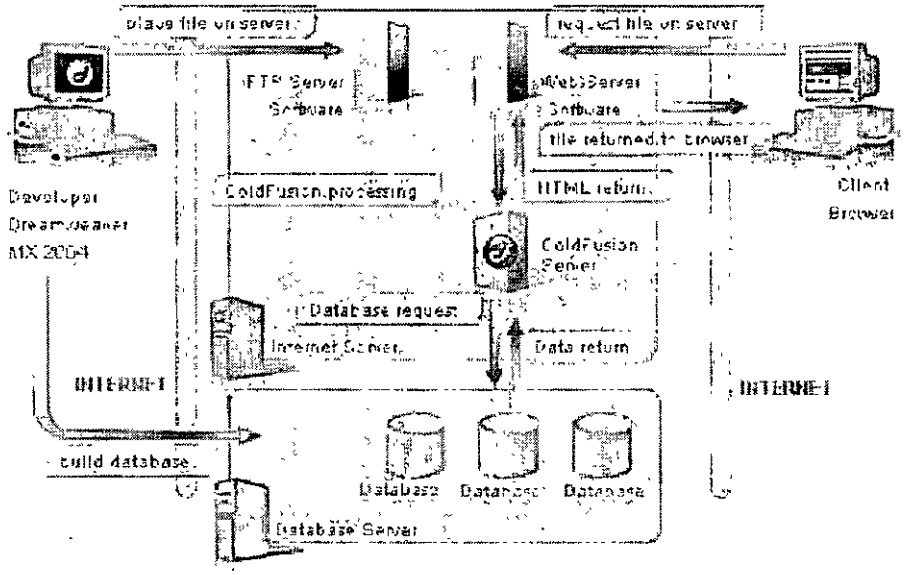
Es escalable. ColdFusion fue diseñado para desarrollar sitios complejos y de alto tráfico. A veces, el problema más grande para un diseñador web es que su sitio se vuelve popular. ColdFusion está diseñado para correr en máquinas multi-procesador, y permite construir sitios que pueden correr en clusters de servidores.

Es un lenguaje server-side. A diferencia de JavaScript y Applets Java, que corren en el cliente o en browsers, ColdFusion corre en el servidor web. Esto significa que los scripts escritos en ColdFusion correrán de la misma manera en cualquier browser.

Componentes de Coldfusion

Tecnología del servidor

En el corazón de cada aplicación de ColdFusion hay un servidor ColdFusion, el cual, combina una arquitectura abierta y extensible que se integra fácilmente con sistemas existentes, así como también con aplicaciones built-in y servicios de infraestructura que ayudan a presentar la información de manera elegante y lograr un alto nivel de desempeño y confiabilidad.



Herramientas de desarrollo

Macromedia ofrece dos herramientas para el desarrollo. Para desarrolladores web, ColdFusion provee una tecnología poderosa de edición de código.

Para diseñadores y desarrolladores web Dreamweaver UltraDev ofrece el mejor ambiente visual de desarrollo. La oferta combinada de ColdFusion 5 y UltraDev 4 Studio, ayuda a mejorar la productividad, asegurar la calidad de la aplicación y diseñar sofisticados sitios web aprovechando la integración con el servidor ColdFusion.

Ambiente de programación

ColdFusion soporta un poderoso lenguaje de scripting en el lado del servidor, ColdFusion Markup Language (CFML), que es extremadamente fácil de aprender y se integra limpiamente con todos los lenguajes y tecnologías web populares. ColdFusion trabaja con múltiples arquitecturas a través de la integración de COM, CORBA y EJB. También puede ser fácilmente extendido con nuevos componentes creados con Java Servlets, clases Java, o C/C++.

Porque usar coldfusion

Según Macromedia:
Permite construir aplicaciones web rápidamente

- Mejora la productividad gracias al lenguaje de scripting del servidor basado en tags, aplicado de manera única en aplicaciones web.
- Acelera el desarrollo con un conjunto poderoso de herramientas poderosas de diseño, programación, depuración e implantación.
- Permite a los equipos de desarrollo colaborar de manera más efectiva compartiendo el mismo servidor y trabajando local o remotamente.

Ensambla soluciones poderosas fácilmente

- El servidor ColdFusion provee funcionalidades built-in como gráficar, seguridad y búsqueda.
- Integración completa con la empresa, se conecta con todo el rango de sistemas backend, incluyendo bases de datos, servidores de mail, directorios, y aplicaciones empaquetadas. Se integra con tecnologías de empresa y de internet, incluyendo COM, CORBA, EJB, XML, C/C++ y Java.
- Posee inteligencia de negocios. Permite crear planillas y reportes tabulares de calidad profesional.
- Completa búsqueda de texto. Permite indexar fácilmente y buscar muchos tipos de contenido, incluyendo páginas web y documentos Microsoft Office 2000.

Entrega un alto desempeño y confiabilidad

- Arquitectura de alto desempeño. Asegura que las aplicaciones sean de implantación multiplataforma, entrega un avanzado thread pooling, caching de páginas built-in, consultas persistentes y pooling de conexiones a bases de datos.
- Administración fácil. Simplifica la implantación y la administración del servidor a través de una poderosa consola de administración basada en web, reportes robustos de servidor y herramientas de análisis, además de integración con los sistemas de administración de la empresa.
- Clustering de servidor. Provee balance de carga y recuperación automática para asegurar que las aplicaciones se mantengan consistentemente disponibles y se escala fácilmente para manejar tráfico creciente.

Ventajas, según CFM resources:

CFML hace fácil la programación web para nuevos desarrolladores, con más de 70 tags CFML y sobre 200 funciones personalizadas, prácticamente cualquier aplicación web puede ser construida rápidamente. ColdFusion puede ser usado en un sitio cada vez que se necesita interacción con el ususario. Procesa formularios, hace seguras algunas partes del sitio, y recolecta o publica datos. Se puede usar para construir diarios murales, clientes de POP mail, calendarios en línea, y salas de chat. Se pueden escribir scripts para rastrear estadísticas.

Usando ColdFusion se ahorra dinero en mantenimiento. Se gasta mucho más dinero en ajustar el software a nuevos requerimientos que en el desarrollo inicial, ColdFusion es experto en el área de mantenimiento sobre otras herramientas middleware para crear sitios web dinámicos, ya que:

Esconde la complejidad, usa menos líneas y son más intuitivas para alcanzar resultados, permite al usuario migrar a otros servidores web y motores de bases de datos con pocos cambios y sin plug-ins externos.

Permite setear y olvidarse de los defaults para el acceso a bases de datos, el programador puede setear los detalles de la conexión a la base de datos una vez y después sólo referirse a la fuente de los datos con un simple nombre. En cambio, con ASP, los detalles de la conexión, como username y password, deben ser repetidos en el código cada vez que se utiliza la fuente de los datos.

Permite setear y olvidarse de la administración de sesiones. Un archivo global, que es transparentemente incluido al comienzo de cada página puede asignar un ID de sesión y un símbolo que hace difícil perder la sesión y puede asociarla transparentemente (através de cookies o URLs) con todas las páginas accesadas por un cliente. La creación de símbolos y la verificación de que el símbolo está amarrado a el ID de la sesión se maneja automáticamente. Otros middleware fuerzan al programador a manejar estos detalles.

ColdFusion simplifica el almacenamiento de variables, el programador puede manipular fácilmente las variables apropiadas a su sesión lógica. Lo mismo pasa con las variables en el servidor, en la aplicacióny al nivel de página. Otros middleware necesitan más compromiso del programador y más trabajo para que sea escalable.

ColdFusion hace loop implícito sobre consultas y listas. Sabe como hacer loop sobre las columnas de una consulta. Obtiene automáticamente la siguiente columna cuando no hay nada más que hacer con la columna actual. Se detiene automáticamente cuando no hay más columnas. Estos detalles no se codifican y no necesitan ser revisados durante el mantenimiento.

ColdFusion genera y envía javascript transparentemente on the fly cuando ciertos tags de input son utilizados. Esto facilita el chequeo de inputs del lado del cliente sin forzar al programador a escribir, revisar y modificar javascript para hacer esto.

ColdFusion usa menos líneas de código y son más intuitivas. supongamos que se necesita desplegar información de una base de datos.

Así se hace con ColdFusion:

```
<cfquery datasource="yourDB">
select VendorID, Vendor from tblVendor order by Vendor
</cfquery>

<cfoutput query="Company">
 #Vendor#, #VendorID#<br>
</cfoutput>
```

Lo mismo en ASP:

```
<%
Option Explicit
Response.Expires = 0
Dim objConn, objRS, strQ
Dim strConnection 14:
Set objConn = Server.CreateObject("ADODB.Connection")
strConnection = "Data Source=somedatasource;"
objConn.Open strConnection
Set objRS = Server.CreateObject("ADODB.Recordset")
Set objRS.ActiveConnection = objConn
strQ = "select VendorID, Vendor "
strQ = strQ & "from Vendor "
strQ = strQ & "order by Vendor"
objRS.Open strQ
%>

<%
While Not objRS.EOF
Response.Write objRS("Vendor") & ", "
Response.Write objRS("VendorID") & "<br>"
objRS.MoveNext
Wend
objRS.close
objConn.close
Set objRS = Nothing
Set objConn = Nothing
%>
```

Lo mismo en JSP

```
<%@ page import="java.sql.*" %>

<%
try {
    Class.forName( "sun.jdbc.odbc.JdbcOdbcDriver" );
} catch (java.lang.ClassNotFoundException e) {
    e.printStackTrace();
}
Connection myConnection = null;
Statement myStatement = null;
ResultSet myResultSet = null;
try {
    myConnection = DriverManager.getConnection ("jdbc:odbc:jsp",
"", "");
    myStatement = myConnection.createStatement();
    myResultSet = myStatement.executeQuery("select VendorID, Vendor
from
    main order by lastname");

    while(myResultSet.next())
    {
        out.println(myResultSet.getString("Vendor")+", ");
        out.println(myResultSet.getString("VendorID")+"<br>");
    }
    myResultSet.close();
    myStatement.close();
    myConnection.close();
} catch (SQLException e) {
    e.printStackTrace();
}
%>
```

Mientras menos líneas haya que revisar y modificar, más dinero se ahorra en mantenimiento.

ColdFusion tiene un buen manejo de errores y depuración. Permite redirigir la información detallada para la depuración a direcciones IP que el programador provea. Cuando un motor de bases de datos arroja un error, ColdFusion sugiere causas posibles. Le permite al programador crear sus propios manejadores de errores cuando se necesita cuidado especial. Con ColdFusion se pueden personalizar los mensajes de error para situaciones específicas y puede proveer un nivele de detalle para los usuarios y uno diferente para los mantenedores.

Las aplicaciones en ColdFusion pueden cambiar de plataformas y motores de bases de datos. Se pueden cambiar las aplicaciones de ColdFusion a diferentes sistemas operativos y servidores web con pequeños cambios, y se puede incluso cambiar los motores de bases de datos con un poco más de esfuerzo. Los proyectos pequeños en ColdFusion pueden crecer sin abandonar código ni empezar del comienzo.

Ventajas inigualables de ColdFusion:

ColdFusion viene con habilidades que otros middleware no pueden alcanzar sin necesitar agregados. Viene con un motor para indexar sitios web. Se puede realizar balance de carga dinámico, se pueden mantener porciones de código como propietarias. Permite agregar componentes de servidores de otros lenguajes, puede usar COM, CORBA y objetos JavaBeans creados por otras herramientas. Se pueden hacer consultas persistentes para mayor velocidad. Se integra con el monitor de desempeño y el monitor de seguridad de NT. Se puede poner a los usuarios en una "caja de arena" (por ejemplo, para soportar múltiples sitios web en un sólo host). Se puede modificar el registro. Permite continuar usando scripts CGI existentes. El código ColdFusion puede ser generado on the fly con herramientas que vienen con su usual editor, ColdFusion Studio. El lenguaje es extensible, convierte datos hacia y desde XML y permite conectar un sitio con otros como en un browser para extender el alvance de la aplicación.

En resumen, según CFM resources:

Dado que la gramática de ColdFusion lleva a menos líneas de código más entendibles, los tiempos de revisión y ajuste se acortan en un tercio o más. Esto gracias a aspectos del lenguaje que esconden la complejidad mientras proveen poder.

Otras herramientas requieren agrupar add ons de distintas fuentes. ColdFusion tiene muchas capacidades built in que otras herramientas no tienen y necesitan apoyarse en add ons.

Otros middleware tienden a encerrar a sus compradores en sistemas operativos específicos, servidores web y motores de bases de datos. Gracias a que ColdFusion está disponible para un amplio rango de sistemas operativos, servidores web y motores de bases de datos, la migración no requiere que los mantenedores empiecen desde el principio.

Sobrepasa a ASP, JSP, Java, etc. Sin embargo, tan importante como su habilidad para soportar comercio web, es su gran retorno sobre la inversión lo que hace elegir a ColdFusion a la hora de mantener comercio en el web.

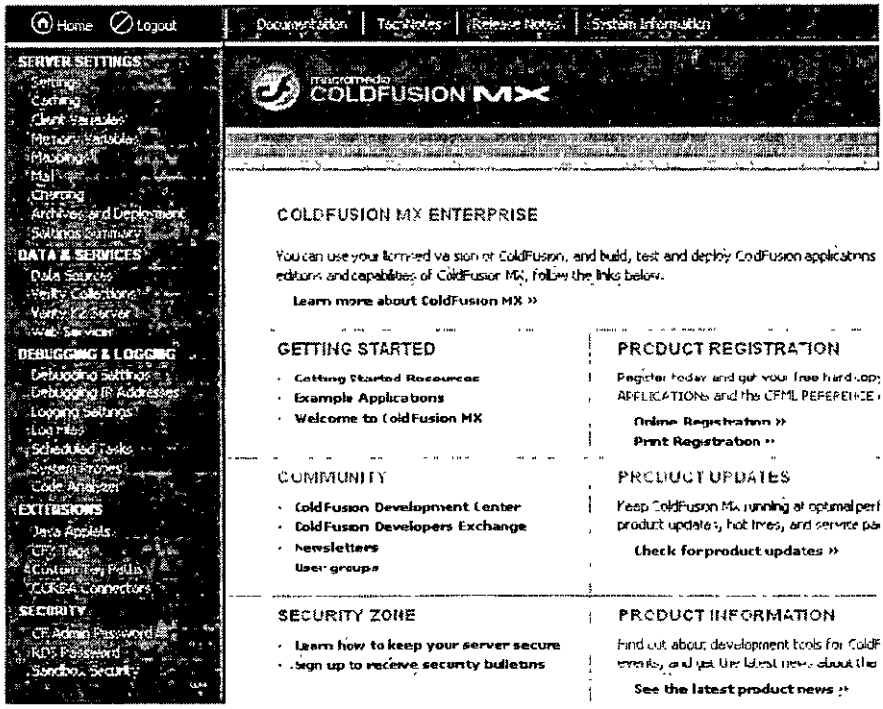
** El siguiente segmento, fue tomado del sitio de Macromedia, el autor es:
Tom Donovan es Escalation Engineer para Macromedia ColdFusion Server

A continuación, encontrará diez recomendaciones para configurar tu servidor de aplicaciones ColdFusion MX y que, toman en cuenta la seguridad. Esta lista está dividida en estas secciones generales:

- Características y valores para desactivar
- Características y valores para activar
- Tareas permanentes

Características y valores para desactivar

1. **Instalación de ColdFusion MX**
   Instale ColdFusion en el servidor de producción sin las aplicaciones de documentación y ejemplo. La opción de Example Applications se puede desactivar durante la instalación del servidor de ColdFusion. Si bien las aplicaciones de ejemplo son útiles para los nuevos desarrolladores de ColdFusion, el código fuente CFML para dichos ejemplos se puede obtener gratuitamente, dando así una oportunidad tentadora a los piratas en un entorno de producción. Las aplicaciones de ejemplo de ColdFusion 5 fueron revisadas para dar mayor seguridad, pero aun así hay pocas razones para tenerlas en un servidor de producción.



2. **Desactive los RDS (Remote Development Services) en su sistema de producción ColdFusion MX**
   La forma más fácil para desactivar los RDS es haciendo que la contraseña RDS en su sistema de producción sea difícil de adivinar. Elija contraseñas que no sean susceptibles de ataques de "diccionario" (palabras que se pueden encontrar en un diccionario). Si no encuentra una contraseña única y difícil de adivinar, una búsqueda rápida en el web con la frase "generador de contraseñas aleatorias" producirá una plétora de ayudas para crear contraseñas. Como regla general, mezcle letras, números y símbolos en sus contraseñas.

3. **Desactive el depurador del servidor de ColdFusion MX**
   En el administrador de ColdFusion MX, puede introducir una lista de direcciones IP para la cual el servidor de ColdFusion desplegará la información de depuración. Esto significa que cuando una máquina con una dirección de IP especificada examina una página CFML, se verá la información de depuración para ese usuario en particular. Observe que si no tiene direcciones IP especificadas y la depuración está activada en el ColdFusion Administrator, la información de depuración puede aparecer para cualquier usuario de la aplicación.

   Para un sistema de producción, introducir una sola dirección IP 127.0.0.1 es una buena idea. Aun en caso de que un pirata ducho parodie esa dirección IP creando paquetes IP con 127.0.0.1 en calidad de dirección IP fuente, dichos paquetes nunca pasarán a través del dominio público de Internet. Algunas direcciones IP no-direccionales son 10.*.*.*, 192.168.*.*, y la gama que va de 172.16.*.* a 172.31.*.*. Si su red interna usa estas direcciones, le resultaría muy seguro poner la dirección IP de algunas de sus máquinas locales en la lista.

Características y valores para activar

1. **Especifique un handler de error que abarque todo el sitio en un sistema de producción**
   En la página Settings de ColdFusion MX Administrator se puede especificar una página de modo que sea el handler de error para todo el sitio. Observe que para ColdFusion, deberá introducir una ruta de acceso relativa (como /swerror.cfm), y no una ruta de acceso absoluta para el archivo. No es necesario que la página que especifique sea muy detallada. Con sólo enviar un mensaje sencillo a su usuario avisándole que ha habido contratiempos, será suficiente. Puesto que la página de error es una página de ColdFusion, se puede producir lógica que le envíe la información sobre el error encontrado por el usuario.

2. **Por ejemplo**, puede hacer que la información sobre el fallo se envíe por correo electrónico al ColdFusion MX Administrator y a la vez mostrar un mensaje genérico al usuario (o sus atacantes) que no contenga ninguna información sobre la página que se está ejecutando. Puede usar la etiqueta cflog, introducida en ColdFusion 5, para registrar el error en las anotaciones de ColdFusion. El error abarcador del sitio debe evitar realizar demasiadas acciones ya que el diseño dejaría mucho que desear si la propia página de ColdFusion del error abarcador del sitio ocasionara un error. En su lugar, se pueden emplear las etiquetas cftry y cfcatch en su handler de error.

3. **Especifique un handler de plantilla faltante que abarque todo el sitio**
Muchas veces se puede especificar la misma página como su handler de error abarcador del sitio. Ni los usuarios legítimos ni los infames piratas tienen por qué saber mucho sobre las páginas faltantes (al menos, no más de lo que les mostraría en su handler de error abarcador del sitio). Usar la misma página ayuda a disuadir y frustar a los piratas que sondean su sitio web en busca de páginas con nombres similares. Se puede recopilar información en abundancia sobre un ataque potencial si se usa la etiqueta cflog. Al usar la misma plantilla para el handler de error abarcador del sitio y el handler de plantilla faltante en todo el sitio se mantendrá al pirata en la oscuridad ya que no sabrá si su sondeo fue estropeado o si simplemente dejó incompletos los datos requeridos para una página de ColdFusion.

4. **Proteja la contraseña de ColdFusion MX Administrator**
Proteja la contraseña de ColdFusion Administrator de la misma manera que protege la contraseña RDS. Todos los servidores web tienen algún mecanismo para restringir los árboles de directorios a usuarios autorizados. Quizá resulte útil usar este mecanismo para el directorio del ColdFusion Administrator también. Recuerde que hay archivos en el árbol de directorios /CFIDE a los cuales usuarios ordinarios necesitan tener acceso. Es preferible restringir el acceso al árbol de directorios /CFIDE/Administrator y no a todo el árbol /CFIDE.

Tareas permanentes

1. **Lea las anotaciones de ColdFusion MX y del servidor web**
Muy pocos desarrolladores leen -literalmente- las anotaciones de ColdFusion y del servidor web, pero es importante que las lea de vez en cuando. Los scripts que se aprovechan de las vulnerabilidades conocidas son fáciles de encontrar en las anotaciones de los servidores web. Al revisar las anotaciones, fíjese si hay una dirección IP única que haya hecho cientos de solicitudes HEAD o solicitudes de nombres de archivos improbables como "CMD.EXE." (No hace mucho, se descubrió que IIS exponía una vulnerabilidad que permitía a los piratas ejecutar CMD.EXE y pasarle una línea de comando. Si tiene un sitio Unix, seguramente crea que este esfuerzo no es demasiado riesgoso, pero tenga en cuenta que se han dado casos impresionantes de explotación de la seguridad en Unix también.)

2. **Lea los boletines de seguridad**
   Lea los boletines de seguridad para todos los productos que usa en su sitio web. La mayoría de las empresas de software cuentan con un área de seguridad. En Macromedia, ésta es www.macromedia.com/security (en inglés) ; en Microsoft, www.microsoft.com/security (en inglés) ; en Sun, www.sun.com/security (en inglés) ; y así por el estilo. El tiempo transcurrido entre el momento en que se descubre una vulnerabilidad de seguridad y el uso generalizado de scripts malintencionados que sondean los sitios web para investigar su vulnerabilidad está reduciéndose. La situación puede fácilmente convertirse en una carrera entre el autor de los scripts malintencionados y los administradores del sitio web. Como lo más seguro es que quiera ganar esta carrera, es mejor registrarse para recibir notificaciones automáticas por correo electrónico de los boletines de seguridad. En la zona de seguridad de Macromedia, regístrese para el Notification Service de la zona de seguridad (en inglés). Normalmente los vendedores de servidores web ofrecen un servicio similar al que se puede suscribir.

3. **Prepare un plan de recuperación**
   Si es objeto de un ataque pirata, tenga un plan de recuperación listo para aplicarse de inmediato. Lo ideal es que nunca lo necesite, pero es difícil concentrarse en reparar un sitio web después de un ataque a la seguridad si no tiene un plan a mano. Si es atacado, hay dos categorías amplias de tareas a realizar:

   - *Conserve la evidencia*: es decir, las anotaciones de su servidor web y las anotaciones de ColdFusion MX. Esto le ayudará a entender cómo fue que el pirata manipuló su sitio web. Si bien es cierto que resulta útil poder actualizar el sitio web con una versión de seguridad, también es útil comparar la versión adulterada de su aplicación con una versión no adulterada de archivos para que se pueda determinar lo que ha sido adulterado. Una copia de seguridad de su base de datos, si es posible, también puede ser evidencia forense útil.

   - *Restaure su sitio web*. Como mínimo, querrá restaurar su aplicación y base de datos al estado no adulterado. Si no puede determinar rápidamente cómo se lanzó el ataque y qué fue lo que afectó, sería útil también reinstalar todo el software, incluyendo el sistema operativo, el servidor web y el servidor ColdFusion MX.

     Si ud. tiene la versión ColdFusion Enterprise, se puede utilizar la utilidad Archive/Deploy para reconfigurar rápida y fácilmente ColdFusion (suponiendo que creó un archivo nuevo después del último cambio hecho). De lo contrario, una copia de seguridad de su aplicación y las anotaciones de las fuentes de datos y otros valores de ColdFusion será suficiente. También sería aconsejable tener a mano registros similares de la configuración de su servidor web y los valores de la base de datos en caso de que tenga que reinstalar todo su software sin previo aviso.

Técnicas de prueba: un ejemplo de JavaScript oculto

Existe una miríada de técnicas para la fase de desarrollo destinadas a crear sitios web a prueba de piratas. De hecho, hay libros enteros que tratan de este tema aunque eso va más allá de esta lista de las diez recomendaciones de seguridad más importantes para aplicar en la fase de producción. Sin embargo, deberá tener siempre en cuenta que la prevención de violaciones a la seguridad también es importante durante el desarrollo. Aun después de pasar su aplicación a producción, deberá hacer pruebas para ver si hay vulnerabilidades. El siguiente es un ejemplo de una técnica conocida como "cross-site scripting."

Este ataque común a sitios web es un ardid para saltarse las reglas de JavaScript. Todos los navegadores modernos se adhieren a esta regla: **JavaScript sólo puede tener acceso a las cookies para el sitio web de donde procede.** El truco está en que un sitio malintencionado dirija a uno de sus usuarios confiados a su propio sitio a través de un vínculo HTML. Este vínculo no sólo contiene el URL para su sitio, sino que también contiene JavaScript solapado, incrustado en un campo de texto. Si su aplicación devuelve este texto al usuario confiado, no aparecerá en la ventana del mismo, si no que se ejecutaría como que proviene de su sitio, y técnicamente es así.

Por ejemplo, el sitio malvado, podría tener un vínculo dirigido a su página de conexión que tenga este código:

```
<a href="http://www.yoursite.com/login.cfm?username=Ralph
Cramden<script>{malicious_code}</script>&password=yyy>
```

Cuando el usuario confiado hace clic en este vínculo, la respuesta será indudablemente la que da su aplicación sobre contraseña no aceptada (a menos que realmente tenga un usuario llamado "Ralph Cramden" y su contraseña sea "yyy"). Mientras la persona se pregunta por qué su contraseña no ha sido aceptada y por qué la ha llamado "Ralph Cramden," la parte del código malintencionado {malicious_code} del script se estará ejecutando en su navegador. El script podría estar recogiendo todas las cookies que pueda y enviándolas a un programa que se encuentre en el sitio web del pirata. Las cookies cosechadas serán las cookies de su sitio web que pertenezcan al usuario, ya que en lo que respecta al navegador del usuario, JavaScript proviene de su sitio.

La solución está en desinfectar cualquier texto introducido por el usuario antes de devolverlo al navegador del usuario. Se pueden usar las funciones HTMLEdit() y XMLEdit() de ColdFusion. Cualquier etiqueta atajaría a cualquier JavaScript incrustado. Esto se logra convirtiendo la etiqueta <script> en <script>. XMLFormat() también cambia el signo de apóstrofo por ' y maneja < y > convirtiéndolos en equivalentes ASCII. Cualquiera de las funciones desarmaría a un script malintencionado y lo forzaría a mostrarse inofensivamente en el navegador del usuario en vez de permitir que se ejecute.

Puesta a prueba de su aplicación

Su aplicación se puede someter a prueba fácilmente para ver si es susceptible de este ataque. Sólo copie el siguiente script no tan malintencionado y trate de pegarlo en varios cuadros para la introducción de texto en su aplicación. Si sólo ve el script en su pantalla, no hay problema, pero si aparece el cuadro de alerta en el navegador, es posible que su aplicación dé lugar a ataques Cross-Site Scripting.

```
<script>alert("Ooops!")</script>
```

Quizá se sorprenda de la frecuencia con que aparece el cuadro de alerta en la pantalla.

Conozca más

URL: http://www.php.net/

URL: http://www.asp.net/

URL: http://www.macromedia.com/es/devnet/coldfusion/

URL: http://www.mysql.com/

Descargar este manual y otros articulos de interés en PDF.
URL: http://www.eurekainweb.com/mineria