

Facultad de Matemáticas

Notas de Curso para la Asignatura
“Programación Web con PHP y MySQL”

Impartida a los alumnos del sexto y octavo
semestre de la Licenciatura en Ciencias de la
Computación, Unidad Tizimín

Elaborado por:
M.C. Cinhtia Maribel González Segura
M.C. Michel García García

*Tizimín, Yucatán.
Enero – Julio 2008*

Contenido

Carta descriptiva	3
Contenido del programa	4
Guía de uso	8
Unidad 1. Introducción	9
1.1 Etiquetas HTML	9
1.2 Formularios y controles	13
1.3 Validación con Javascript	14
1.4 Estándares de calidad	30
1.5 Estilos CSS	33
Unidad 2. Servidores Web, PHP y MySQL	41
2.1 Características de un servidor Web	41
2.2 Instalación y configuración de un Servidor Web Apache	43
2.3 Instalación y configuración del servidor PHP	49
2.4 Instalación y configuración de MySQL	55
2.5 Administración de bases de datos MySQL	57
2.6 Aplicaciones integrales APACHE + MYSQL + PHP	79
Unidad 3. Scripts en PHP	80
3.1 Fundamentos de PHP	80
3.2 Declaración de variables y constantes	81
3.3 Declaración de funciones	86
3.4 Acceso a datos enviados a través de una página Web	88
3.5 Generación de contenido dinámico	92
3.6 Acceso a archivos	92
Unidad 4. Creación de aplicaciones	94
4.1 Definición de la aplicación	94
4.2 Diseño de la interfaz	95
4.3 Validación de los formularios	96
4.4 Implementación de la aplicación	97
Unidad 5. Publicación en la Web	98
5.1 Requerimientos básicos	98
5.2 Recursos gratuitos	99
5.3 Administración remota	99
5.4 Mantenimiento de un sitio web	100
Bibliografía	101

Carta descriptiva

La asignatura “Programación Web con PHP y MySQL” se imparte como optativa a los estudiantes del sexto y octavo semestre de la Licenciatura en Ciencias de la Computación, plan 2004. El presente material está elaborado con base en los contenidos de dicha asignatura.

El material que aquí se presenta fue elaborado por la M.C. Cinhtia M. González Segura y el M.C. Michel García García para el semestre Enero – Julio 2008, y se distribuyó a los alumnos en su momento, durante el curso.

En primer lugar se incluye el programa del curso, después se presenta una guía de uso para los recursos presentados, se incluyen las notas del curso por unidades, las cuales pueden ser complementadas con el paquete didáctico creado por los mismos autores. El contenido de la unidades se estructura con instrucciones y ejemplos de código que pueden ser fácilmente implementados por los estudiantes.

Datos de la Asignatura

Clave: OP-09

Nombre: Programación web con PHP y MySQL

Plan: Licenciatura en Ciencias de la Computación (LCC), Ingeniería de Software (IS)

Semestre: Quinto en adelante.

Créditos: 10

Hrs semanales: 4

Total de horas: 72

Requisitos previos: Programación

Contenido del programa

UNIVERSIDAD AUTÓNOMA DE YUCATÁN FACULTAD DE MATEMÁTICAS

MISIÓN

Formar profesionales altamente capacitados, desarrollar investigación y realizar actividades de extensión, en Matemáticas y Computación, así como en sus diversas aplicaciones.

LICENCIATURA EN CIENCIAS DE LA COMPUTACIÓN

OBJETIVO:

El alumno desarrollará aplicaciones accesibles a través de páginas Web dinámicas, empleando tecnologías actuales en la solución de problemas particulares.

MATERIA	:	PROGRAMACIÓN WEB CON PHP Y MYSQL
NIVEL	:	SEXTO Y OCTAVO SEMESTRE
PERIODO	:	ENERO – JULIO 2008
PROFESORES	:	M.C. CINHITIA MARIBEL GONZÁLEZ SEGURA M.C. MICHEL GARCÍA GARCÍA

FECHA DE ELABORACIÓN: Enero 2008

AUTORES: M.C. Michel García García, M.C. Cinhtia Maribel González Segura

DURACIÓN DEL CURSO: 90 horas.

DESCRIPCIÓN DE LA ASIGNATURA

PHP se ha convertido en un nuevo estándar dentro de las Tecnologías de Información por diversos motivos, al ser un lenguaje optimizado y creado para Internet, además de ser multiplataforma y tener un bajo costo de licenciamiento e implementación.

Este curso fue diseñado con el fin de incluir herramientas actuales de utilidad para los estudiantes se desempeñarán en el área laboral relacionada con el desarrollo de aplicaciones.

Durante el curso se realizará el diseño y elaboración de páginas Web dinámicas que incluyan formularios para interactuar con una base de datos MySQL a través de scripts programados en PHP. Se propone desarrollar y publicar el sitio final en un servidor gratuito de Internet.

RECURSOS DIDÁCTICOS

Laptop y proyector
Sala de cómputo
Pizarrón

CONTENIDO

Unidad 1. Introducción

15 hrs

Objetivo: *Crear páginas Web empleando correctamente las etiquetas HTML y las hojas de estilos CSS según el estándar del W3C.*

- 1.1 Etiquetas HTML
- 1.2 Formularios y controles
- 1.3 Validación con javascript
- 1.4 Estándares de calidad
- 1.5 Hojas de estilo CSS

Unidad 2. Servidores: Web, Php y Mysql

20 hrs

Objetivo: *Instalar y configurar adecuadamente los servidores de páginas Web, Php y MySQL para un correcto funcionamiento de las aplicaciones cliente-servidor.*

- 2.1 Características de un servidor web.
- 2.2 Instalación y configuración de un servidor Web Apache
- 2.3 Instalación y configuración del servidor PHP
- 2.4 Instalación y configuración de MySQL
- 2.5 Administración de bases de datos MySQL
- 2.6 Aplicaciones integrales APACHE + MYSQL + PHP

Unidad 3. Scripts en PHP

30 hrs

Objetivo: *Codificar correctamente los scripts en el lenguaje PHP que permitan la elaboración de aplicaciones en línea.*

- 3.1 Fundamentos de PHP
- 3.2 Declaración de variables y constantes
- 3.3 Declaración de funciones
- 3.4 Acceso a datos enviados desde una página Web
- 3.5 Generación de contenido dinámico
- 3.6 Acceso a archivos

Unidad 4. Creación de aplicaciones

15 hrs

Objetivo: *Diseñar la interfaz gráfica con formularios que permitan la sencilla utilización de una aplicación en línea.*

- 4.1 Definición de la aplicación
- 4.2 Diseño de la interfaz
- 4.3 Validación de los formularios
- 4.4 Implementación de la aplicación

Unidad 5. Publicación en la Web

10 hrs

Objetivo: *Publicar y administrar un sitio Web en un servidor remoto.*

- 5.1 Requerimientos básicos
- 5.2 Recursos gratuitos
- 5.3 Administración remota
- 5.4 Mantenimiento de un sitio web

PROYECTO SUGERIDO

Se sugieren dos enfoques de proyectos a elegir:

- Sitio para inscripciones remotas. Orientado al área que el estudiante desee, con la característica de que permita la inscripción a un evento específico.
- Gestión de catálogos, con autenticación de usuarios. El tema de los catálogos es flexible, pero requerirá realizar la administración de las tablas.

CRITERIOS DE EVALUACIÓN

2 Exámenes parciales: 60 %

Prácticas: 10 %

Tareas y participación: 10%

Proyecto final: 20 %

Los 2 exámenes parciales consisten en la entrega de los avances de su proyecto. La primera entrega se realizará después de la primera unidad y consiste en el diseño de su sitio web con las secciones que éste incluirá. La segunda entrega se realizará al finalizar la unidad 3 y consiste en la implementación de su aplicación en PHP. El proyecto final debe contener las validaciones de su formulario y la publicación de su sitio en Internet. Las prácticas son realizadas durante la clase.

METODOLOGÍA DE ENSEÑANZA

Exposición, Interrogatorio, Práctica, Tareas (individual y por equipo)

El maestro expone los conceptos fundamentales de los temas en cuestión, da ejemplos y por último motiva al estudiante a resolver ejercicios en clase. Para reforzar la comprensión de estos ejercicios se dejarán tareas que consistirán en el diseño de la solución de los problemas asignados y algunas veces de la programación de esta solución.

SOFTWARE REQUERIDO

Dreamweaver CS3 (o alguna versión similar)

Servidor Apache (gratuito, se puede obtener en <http://www.apache.org/>)

Servidor PHP (gratuito, se puede obtener en <http://www.php.net/>)

Servidor MySQL (gratuito, se puede obtener en <http://www.mysql.com/>)

REQUISITOS PREVIOS

Bases de datos

Conocimientos básicos de html

Deseable conocimiento básico de mysql

BIBLIOGRAFÍA

1. César Pérez. **Macromedia Dreamweaver MX 2004, desarrollo de páginas web dinámicas con PHP y MySQL**. Alfaomega, RA-MA. 2004.
2. López Quijado, José. **Domine PHP y MySQL. Programación dinámica en el lado del servidor**. Editorial Ra-ma. Madrid, 2007.
3. Cabezas Granado, Luis Miguel. **PHP 5**. Anaya Multimedia. 2004.
4. Gil Rubio, Javier; Yagüe Panadero, Agustín; Tejedor Cerbel, Jorge; Alonso Villaverde, Santiago. **Creación de sitios web con PHP 5**. McGraw-Hill.
5. Powers, David. **Desarrollo web dinamico con dreamweaver 8 y php (diseño y creatividad)**. Anaya Multimedia-Anaya Interactiva.
6. Tobias Ratschiller, Till Gerken. **Creación de aplicaciones Web con PHP 4**. Editorial Prentice Hall, Madrid, 2001.
7. Tim Converse, Joyce Park, **PHP Bible, 2nd Edition**, Willey Publishing, Inc.
8. Chris Lea, Mike Buzzard, Dilip Thomas, Jessey White-Cinis, **PHP MySQL Website Programming: Problem - Design - Solution**, Editorial Apress.
9. Luke Welling, Laura Thomson, **PHP and MySQL Web Development**, Second Edition, SAMS, 2003.
10. Andy Harris, **PHP/MySQL Programming for the Absolute Beginner**, Learning Express, 2003.

REFERENCIAS

1. <http://www.w3.org/MarkUp/Guide/>
2. <http://www.w3.org/Style/Examples/011/firstcss.es.html>
3. <http://www.webestilo.com/>
4. <http://www ldc.usb.ve/~vtheok/webmaestro/>
5. <http://www.lsi.us.es/cursos/cursophp/>
6. <http://www.programacion.net/tutorial/php/>
7. http://es.tldp.org/Manuales-LuCAS/manual_PHP/manual_PHP/
8. <http://www.desarrolloweb.com/manuales/6/>

PERFIL PROFESIOGRÁFICO DEL PROFESOR

Licenciado en Ciencias de la Computación o carrera afín, preferentemente con estudios de posgrado y experiencia docente, de investigación o de trabajo en el área.

Guía de uso

Se presenta el material teórico y práctico necesario para el estudio de la asignatura. Las unidades están estructuradas de acuerdo al contenido programático que se presenta en la sección anterior y para un uso más eficiente se sugiere consultar también el paquete didáctico realizado por los mismos autores, el cual se acompaña de un CD con los códigos fuentes de ejemplos específicos que se describen de manera más general en estas notas del curso.

La primera unidad se sugiere distribuirla al inicio del curso junto con el programa de la asignatura, para que los estudiantes que no hayan desarrollado páginas web se familiaricen con el lenguaje HTML y así dedicarle un poco más de tiempo a los estilos CSS, por ser un poco más recientes.

La segunda unidad sirve como guía en el proceso de instalación y se sugiere que los estudiantes con menos experiencia en la instalación y configuración de sistemas lo consulten por cualquier conflicto que pudiera surgir durante la instalación.

La Unidad 3 contiene la parte más importante del curso pues es en sí el lenguaje PHP. Se sugiere distribuir a los estudiantes al principio de la unidad y referirse a esta parte cuando se realicen las prácticas y actividades en clase.

Las Unidades 4 y 5 son un complemento para poder finalizar exitosamente una aplicación completa de tal forma que sea usable, se describe el proceso completo para crear una aplicación y las características que se deben considerar antes y durante el desarrollo.

Unidad 1. Introducción

Internet es una red formada por cientos de miles de millones de computadoras interconectadas que se encuentran distribuidas en todo el mundo.

En gran parte, el espectacular crecimiento de Internet se debe a la notable mejora en la facilidad de uso de los servicios ofrecidos, dado que, aún manteniéndose los servicios originales de transferencia de archivos, correo electrónico o acceso remoto. Además, la World Wide Web o WWW (servicio de consulta de documentos hipertextuales) ha sido una pieza clave para que la Internet posea la popularidad de la que actualmente goza.

En esta unidad se describe la estructura y etiquetas básicas para la creación de un sitio web, así como los elementos necesarios para validar los campos incluidos en un formulario y algunos conceptos fundamentales para desarrollar páginas Web, respetando los estándares de desarrollo actuales, incluyendo el uso de hojas de estilo.

1.1 Etiquetas HTML

HTML es un “Lenguaje” para marcado de hipertexto (HyperText Markup Language), aunque estrictamente hablando no es un lenguaje sino un conjunto de etiquetas empleadas para dar formato al texto que posteriormente será visualizado a través de algún navegador [12]. La estructura básica de un documento HTML es la siguiente:

```
<HTML>
<HEAD>
<TITLE>Ejemplo 1</TITLE>
</HEAD>
<BODY>
Hola mundo
</BODY>
</HTML>
```

El código anterior puede ser guardado como un documento de texto (en un editor común) con la extensión .htm o .html. Posteriormente, al visualizarlo con algún navegador se obtendrá como resultado la frase “Hola Mundo”, para no romper la tradición y buena costumbre al escribir la primera aplicación en cualquier entorno de programación.

Si se emplea algún editor de texto especializado en la creación de páginas web (como el Dreamweaver, en cualquiera de sus versiones) no será necesario escribir letra a letra cada una de las etiquetas, aunque es una buena costumbre conocerlas y saber para qué sirve cada una de ellas. Cabe mencionar que HTML no es sensible a mayúsculas y minúsculas, por lo que escribir <html> produce el mismo efecto que escribir <HTML>.

A continuación se listan las principales etiquetas HTML:

Etiqueta	Descripción
<html></html>	Al principio y al final de todo documento, se usa una sola vez.
<head></head>	Cabecera del documento. Dentro del head se crean las etiquetas title, meta y opcionalmente las de lenguajes scripts o estilos.

<title> </title>	Indica el título de la página para el navegador.
<meta>	permite aportar metainformación al documento, para su mejor identificación e indexación por los motores de búsqueda. Ejemplos: <meta name="description" content="Frase descriptiva de los contenidos de la página"> <meta name="keywords" content="Palabras clave que resuman la temática de los contenidos de la página"> <meta name="author" content="Nombre del autor de la página">
<body> </body>	Dentro de esta etiqueta se insertan los contenidos del documento El cierre de la etiqueta </body> se coloca justo antes del cierre </html>

Además, cada etiqueta posee sus propios atributos, que permitirán modificar la apariencia de la misma, al ser aplicada sobre el texto afectado. Por ejemplo, la etiqueta <body> puede incluir información referente a las propiedades de la página, usando los siguientes atributos:

bgcolor	define el color de fondo de la página
text	define el color por defecto del texto en la página
link	define el color de los enlaces
vlink	el color de los enlaces visitados
alink	define el color de los enlaces activos
background	establece una imagen para el fondo de la página. <body background = "yo.gif">

Todos estos parámetros se pueden agrupar en una única etiqueta <body>:

```
<body bgcolor="#xyyyzz" text="#xyyyzz" link="#xyyyzz" vlink="#xyyyzz"
alink="#xyyyzz">
```

Donde xx es un código hexadecimal indicando el nivel de Rojo, yy indica el nivel de Verde y zz el nivel de Azul.

Además, existen las etiquetas <!-- texto de comentarios --> que permiten anotar aclaraciones privadas del autor de la página. Lo que se escribe dentro de esta etiqueta es ignorado por el navegador, no se visualiza en la página.

Las etiquetas que permiten añadir formato al texto son las siguientes:

 	Texto en negrita. También sirve la etiqueta ...
<i> </i>	Texto en cursiva. También sirve la etiqueta ...
<u> </u>	Texto subrayado
 	Tamaño del texto, X es un valor de 1 a 7, o un valor relativo (+1 a +7 y -1 a -7)
 	Color del texto, donde XXYYZZ es un hexadecimal indicando el color
 	Tipo de la fuente
<pre>	preformateado. Respeta espacios, saltos de línea y los retornos utilizados

La etiqueta puede incluir los tres parámetros (tamaño, fuente y color):
 Texto formateado

De manera similar, para agregar formato a los párrafos se dispone de las siguientes etiquetas:

<p> salto de párrafo </p>
salto de línea

<blockquote> texto con sangría </blockquote>
<center> texto centrado </center>
<p align=center> párrafo centrado </p>
<p align=left> párrafo alineado a la izquierda</p>
<p align=right> párrafo alineado a la derecha</p>

Si se requiere emplear listas en el texto, se dispone de las siguientes:

 Inicia lista no numerada
 elemento de la lista
 termina lista

 Inicia lista numerada
 elemento de la lista
 termina lista

<dl> Inicia lista de glosario o definición
<dt> término por definir</dt>
<dd> definición del término</dd>
</dl> termina lista

Para incluir líneas horizontales de separación:

<hr> línea horizontal
<hr width="x%"> anchura de la línea en porcentaje
<hr width=x> anchura de la línea en píxeles
<hr size=x> altura de la línea en píxeles
<hr align=center> línea alineada en el centro
<hr align=left> línea alineada a la izquierda
<hr align=right> línea alineada a la derecha
<hr noshade> línea sin efecto de sombra

Para la inserción de imágenes:

 indica la ruta y nombre de la imagen (.gif o .jpg)
 borde de grosor X en torno a la imagen
 tamaño de la imagen (alto y ancho) en píxeles
 texto al pasar el cursor sobre la imagen
 alineación inferior del texto respecto de la imagen
 alineación del texto en el centro de la imagen
 alineación superior del texto respecto de la imagen
 alineación izquierda de la imagen en el párrafo
 alineación derecha de la imagen en el párrafo
 espacio horizontal entre la imagen y el texto

`` espacio vertical entre la imagen y el texto

Para la inserción de tablas, útiles para organizar la página y presentar datos tabulares:

`<table>.....</table>` Define inicio y fin de la tabla
`<table width="XX%">` Determina el ancho de la tabla, en píxeles o porcentaje.
`<table height="XX">` Determina la altura de la tabla en píxeles
`<table border="X">` Establece el grosor en píxeles del borde de la tabla
`<table cellpadding="X">` Define el espacio en píxeles entre las celdas
`<table cellspacing="X">` Define el espacio en píxeles entre el borde y el texto
`<tr>.....</tr>` determina cada una de las filas de la tabla
`<td>.....</td>` determina cada una de las columnas dentro de las filas
`<td rowspan="2">` Texto `</td>` une dos celdas de filas adyacentes, en una celda.
`<td colspan="2"> </td>` une dos celdas de columnas adyacentes en una celda.

Ejemplo de tabla de 2 filas y 3 columnas

```
<table width="100%" height="200" border="1" cellpadding="5">
<tr>
<td>primera columna de la fila 1</td>
<td>segunda columna de la fila 1</td>
<td>tercera columna de la fila 1</td>
</tr>
<tr>
<td>primera columna de la fila 2</td>
<td>segunda columna de la fila 2</td>
<td>tercera columna de la fila 2</td>
</tr>
</table>
```

Dentro de cada celda se puede alinear el texto o cualquier contenido, cambiar el color de fondo, con las etiquetas anteriores para texto, párrafos o imágenes.

Para crear enlaces entre páginas:

``Enunciado del enlace``
`` Vínculo a una dirección de correo electrónico.
`` Define un marcador (ancla) en un punto concreto de una página, para poder enlazarlo posteriormente.
`` dirige un enlace interno al punto dónde está el marcador.
`` enlace a un punto concreto de otra página.

Atributos adicionales de la etiqueta `<a>` (anchor):

`target="_blank"` Abre la página en un nuevo navegador
`target="_top"` Abre la página en una nueva ventana
`title="texto descriptivo del enlace"` descripción del destino del enlace

1.2 Formularios y controles

Los formularios permiten solicitar y enviar información al visitante, por lo que resultan indispensables para añadir dinamismo a un sitio web. Un formulario puede estar compuesto por tantos campos como datos se desee obtener. Una vez introducidos los valores en los controles o campos, la información será enviada a una URL donde se procesará y/o almacenará, según se desee.

La declaración de un formulario se hace por medio de las etiquetas `<form>.....</form>` y dentro de ellas se incluyen los controles que permitirán la recaudación de datos. Los atributos de esta etiqueta son los siguientes:

<code>action=""</code>	Indica la dirección de la página que va a tratar las variables enviadas, un guión CGI o la URL <i>mailto</i>
<code>Method=""</code>	Indica el método de transferencia de las variables. POST, si se envía a través del stdio. GET, si se envía a través de la URL.

La estructura básica de un documento HTML que incluye un formulario (sin una página que lo procese) es la siguiente:

```
<HTML>
<HEAD>
<TITLE>Ejemplo 14</TITLE>
</HEAD>
<BODY>

<H1>Formularios</H1>

<FORM ACTION="" METHOD="POST">
<INPUT TYPE="text" NAME="nombre" /><BR />
<INPUT TYPE="submit" /> <INPUT TYPE="Reset" />
</FORM>

</BODY>
</HTML>
```

Los campos de entrada son los elementos que permiten que un formulario envíe la información que introduzca el usuario.

En el ejemplo anterior se incluyeron dos campos de texto y dos botones, a continuación se describen los atributos de la etiqueta `<input>`

<code>type=""</code>	Indicará el tipo de variable a introducir, que puede ser: text, password, checkbox, radio, hidden.
<code>Name=""</code>	Indicará el nombre que se asigna a un determinado campo
<code>text</code>	Indica que el campo a introducir será un texto. Sus atributos pueden ser: <code>maxlength=""</code> Su valor limita el número máximo de caracteres a introducir en ese campo

	<code>size=""</code> Su valor que limita el número de caracteres que se despliegan en ese campo <code>value=""</code> Indica el valor inicial del campo
<code>Password</code>	Indica que el campo a introducir será una palabra de paso. Mostrará asteriscos en lugar de letras escritas. Sus atributos son los mismos que para <code>text</code>
<code>Checkbox</code>	El campo se elegirá marcando de entre varias opciones una casilla cuadrada. <code>value=""</code> Entre comillas se indicará el valor de la casilla. <code>Checked = "checked"</code> Si se incluye, la casilla aparecerá marcada por defecto.
<code>Radio</code>	El campo se elegirá marcando de entre varias opciones una casilla circular. <code>value=""</code> Entre comillas se indicará el valor de la casilla
<code>hidden</code>	El visitante no puede modificar su valor ya que no está visible. Se manda siempre junto al atributo <code>value=</code> seguido de su valor entre comillas

1.3 Validación con Javascript

JavaScript es el lenguaje que nos permite interactuar con el navegador de manera dinámica y eficaz, proporcionando a las páginas web dinamismo y vida. Se trata de un lenguaje de tipo script compacto, basado en objetos y guiado por eventos diseñado específicamente para el desarrollo de aplicaciones cliente-servidor dentro del ámbito de Internet.

JavaScript es sensible a mayúsculas y minúsculas, todos los elementos de JavaScript deben referenciarse cómo se definieron, no es lo mismo "Salto" que "salto"

Existen distintos modos de incluir lenguaje JavaScript en una página. La forma mas común de hacerlo es utilizando la directiva `<script>` en un documento HTML (se pueden incluir tantas directivas `<script>` como se quiera en un documento). El formato es el siguiente:

```
<script language="Javascript 1.3">
```

El atributo lenguaje hace referencia a la versión de JavaScript que se va a utilizar en dicho script. Otro atributo de la directiva script es `src`, que puede usarse para incluir un archivo externo que contiene JavaScript y que quiere incluirse en el código HTML.

```
<script language="JavaScript" src ="archivo.js">
</script>
```

El archivo externo simplemente es un archivo del texto que contiene código JavaScript, y cuyo nombre acaba con la extensión `js`.

Puede incluirse también código JavaScript como respuesta a algún evento:

```
<input type="submit" onclick="alert('Acabas de hacer click');return false;" value="Click">
```

Variables en JavaScript

Las variables almacenan y recuperan datos, también conocidos como "valores". Una variable puede referirse a un valor que cambia o se cambia. Las variables son referenciadas

por su nombre, y el nombre que les es asignado debe ser conforme a ciertas reglas (debe empezar con una letra o ("_")); los caracteres siguientes pueden ser números (0-9), letras mayúsculas o letras minúsculas).

Las variables en JavaScript pueden ser de alcance global o local. Una variable global es accesible desde cualquier <script> de la página mientras que una variable local sólo lo es desde la función en la que fue declarada.

Normalmente, se crea una nueva variable global asignándole simplemente un valor:

```
globalVariable=5;
```

Sin embargo, si se está codificando dentro de una función y se quiere crear una variable local que sólo tenga alcance dentro de esa función, se debe declarar la nueva variable haciendo uso de var:

```
function newFunction()
{
  var localVariable=1;
  globalVariable=0;
  ...
}
```

Tipos de Datos

JavaScript tiene la peculiaridad de ser un lenguaje débilmente tipado, esto es, una variable puede cambiar de tipo durante su vida, por ejemplo uno puede declarar una variable que ahora sea un entero y más adelante una cadena.

```
MiVariable=4;
```

y después:

```
MiVariable="Una_Cadena";
```

Un ejemplo de página web que incluye un script de javascript es el siguiente:

```
<!-- Manual de JavaScript -->
<html>
<head>
  <title>Ejemplo de JavaScript</title>
</head>
<body>
<script language="JavaScript">
  a = 8;
  b = 3;
  document.write(a + b);
  a++;
  document.write(a);
</script>
```

```
</body>  
</html>
```

A diferencia de otros lenguajes, en Javascript no es necesario declarar las variables especificando el tipo de dato que contendrán, será el propio interprete el que le asignará el tipo apropiado. (Esto es así para seguir la filosofía de diseño de Javascript que indica que se realizan programas pequeños y que la idea es lograr que el programador realice los scripts de la manera más rápida posible).

Los tipos de datos en Javascript son los siguientes:

Números	Enteros o coma flotante.
Booleanos	True o False.
Cadenas	Los tipos de datos cadena deben ir delimitados por comillas simples o dobles.
Objetos	Obj = new Object();
Nulos	Null
Indefinidos	Un valor indefinido es el que corresponde a una variable que ha sido creada pero no le ha sido asignado un valor.

Operadores Aritméticos

Los operadores aritméticos toman los valores numéricos (literales o variables) como sus operando y devuelve un solo valor numérico. Los operadores aritméticos normales son:

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números
-	Substracción	7 - 9	Resta dos números
*	Multiplicación	6 * 3	Multiplica dos números
/	División	4 / 8	Divide dos números
%	Módulo: el resto después de la división	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Incremento.	a++	Suma 1 al contenido de una variable.
--	Decremento.	a--	Resta 1 al contenido de una variable.

-	Invierte el signo de un operando.	-a	Invierte el signo de un operando.
---	-----------------------------------	----	-----------------------------------

Operadores relacionales

Un operador relacional compara sus operando y devuelve un valor lógico que depende de si la comparación es verdad o no. Los operando pueden ser numéricos o cadenas.

Operador	Descripción
==	" Igual a" devuelve true si los operandos son iguales
===	Estrictamente "igual a" (JavaScript 1.3)
!=	" No igual a" devuelve true si los operandos no son iguales
!==	Estrictamente " No igual a" (JavaScript 1.3)
>	" Mayor que" devuelve true si el operador de la izquierda es mayor que el de la derecha.
>=	" Mayor o igual que " devuelve true si el operador de la izquierda es mayor o igual que el de la derecha.
<	" Menor que" devuelve true si el operador de la izquierda es menor que el de la derecha.
<=	"Menor o igual que" devuelve true si el operador de la izquierda es menor o igual que el de la derecha.

Operadores lógicos

Los operadores Lógicos se utilizan para combinar múltiples comparaciones en una expresión condicional. Un operador lógico toma dos operandos cada uno de los cuales es un valor true o false y devuelve un valor true o false.

Operador	Descripción
&&	" Y " Devuelve true si ambos operadores son true.
	" O " Devuelve true si uno de los operadores es true.
!	"No" Devuelve true si la negación del operando es true.

Operadores de Asignación

El operador de asignación '=' le permite asignar un valor a una variable.

Operador	Descripción
=	Asigna el valor del operando de la derecha a la variable de la izquierda. Ejemplo: inttotal=100;
+= (tambien -=, *=, /=)	Añade el valor del operando de la derecha a la variable de la izquierda. Ejemplo: inttotal +=100
&= (también =)	Asigna el resultado de (operando de la izquierda & operando de la derecha) al operando de la izquierda

Operadores de Especiales

Varios operadores de JavaScript, es difícil clasificarlos en una categoría en particular. Estos operadores se resumen a continuación.

Operador	Descripción
(condición) ? trueVal : falseVal	Asigna un valor especificado a una variable si la condición es true, por otra parte asigna un valor alternativo si la condición es false.
New	El operador new crea una instancia de un objeto.
This	La palabra clave 'this' se refiere al objeto actual.
,	El operador ',' evalúa los dos operados.
Delete	El operador delete borra un objeto, una propiedad de un objeto, o un elemento especificado de un vector.
Void	El operador Void especifica una expresión que será evaluada sin devolver ningún valor.
Typeof	Devuelve el tipo de dato de un operando.

Objetos

Una primera clasificación del modelo de objetos lo dividiría en dos grandes grupos. Por una parte, tendríamos los objetos directamente relacionados con el navegador y las posibilidades de programación HTML (denominados, genéricamente, objetos del navegador) y por otra parte un conjunto de objetos relacionados con la estructura del lenguaje, llamados genéricamente objetos del lenguaje.

El Objeto String

Este objeto nos permite hacer diversas manipulaciones con las cadenas, para que trabajar con ellas sea más sencillo. Cuando asignamos una cadena a una variable, JS está creando un objeto de tipo String que es el que nos permite hacer las manipulaciones.

Propiedades

- `length`. Valor numérico que nos indica la longitud en caracteres de la cadena dada.
- `prototype`. Nos permite asignar nuevas propiedades al objeto String.

Métodos

- `anchor(nombre)`. Crea un enlace asignando al atributo NAME el valor de 'nombre'. Este nombre debe estar entre comillas " "
- `big()`. Muestra la cadena de caracteres con una fuente grande.
- `blink()`. Muestra la cadena de texto con un efecto intermitente.
- `charAt(indice)`. Devuelve el carácter situado en la posición especificada por 'indice'.
- `fixed()`. Muestra la cadena de caracteres con una fuente proporcional.
- `fontcolor(color)`. Cambia el color con el que se muestra la cadena. La variable color debe ser especificada entre comillas: " ", o bien siguiendo el estilo de HTML, es decir "#RRGGBB" donde RR, GG, BB son los valores en hexadecimal para los colores rojo, verde y azul, o bien puede ponerse un identificador válido de color entre comillas. Algunos de estos identificadores son "red", "blue", "yellow", "purple", "darkgray", "olive", "salmon", "black", "white", ...
- `fontsize(tamaño)`. Cambia el tamaño con el que se muestra la cadena. Los tamaños válidos son de 1 (más pequeño) a 7 (más grande).
- `indexOf(cadena_buscada, indice)`. Devuelve la posición de la primera ocurrencia de 'cadena_buscada' dentro de la cadena actual, a partir de la posición dada por 'indice'. Este último argumento es opcional y, si se omite, la búsqueda comienza por el primer carácter de la cadena.
- `italics()`. Muestra la cadena en cursiva.
- `lastIndexOf(cadena_buscada, indice)`. Devuelve la posición de la última ocurrencia de 'cadena_buscada' dentro de la cadena actual, a partir de la posición dada por 'indice', y buscando hacia atrás. Este último argumento es opcional y, si se omite, la búsqueda comienza por el último carácter de la cadena.
- `link(URL)`. Convierte la cadena en un vínculo asignando al atributo HREF el valor de URL.
- `small()`. Muestra la cadena con una fuente pequeña.

- `split(separador)`. Parte la cadena en un array de caracteres. Si el carácter separador no se encuentra, devuelve un array con un sólo elemento que coincide con la cadena original. A partir de NS 3, IE 4 (JS 1.2).
- `strike()`. Muestra la cadena de caracteres tachada.
- `sub()`. Muestra la cadena con formato de subíndice.
- `substring(primer_Indice,segundo_Indice)`. Devuelve la subcadena que comienza en la posición 'primer_Indice + 1' y que finaliza en la posición 'segundo_Indice'. Si 'primer_Indice' es mayor que 'segundo_Indice', empieza por 'segundo_Indice + 1' y termina en 'primer_Indice'. Si hacemos las cuentas a partir de 0, entonces es la cadena que comienza en 'primer_Indice' y termina en 'segundo_Indice - 1' (o bien 'segundo_Indice' y 'primer_Indice - 1' si el primero es mayor que el segundo).
- `sup()`. Muestra la cadena con formato de superíndice.
- `toLowerCase()`. Devuelve la cadena en minúsculas.
- `toUpperCase()`. Devuelve la cadena en minúsculas.

```
<!-- Manual de JavaScript -->
<HTML>
<HEAD>
  <title>Ejemplo de JavaScript</title>
</HEAD>
<BODY>
<script LANGUAGE="JavaScript">
<!--
var cad = "Hello World",i;
var ja = new Array();

ja = cad.split("o");

with(document) {
write("La cadena es: "+cad+"<BR>");
write("Longitud de la cadena: "+cad.length+"<BR>");
write("Haciendola ancla: "+cad.anchor("b")+"<BR>");
write("En grande: "+cad.big()+"<BR>");
write("Parpadea: "+cad.blink()+"<BR>");
write("Caracter 3 es: "+cad.charAt(3)+"<BR>");
write("Fuente FIXED: "+cad.fixed()+"<BR>");
write("De color: "+cad.fontcolor("#FF0000")+"<BR>");
write("De color: "+cad.fontcolor("salmon")+"<BR>");
write("Tamaño 7: "+cad.fontSize(7)+"<BR>");
write("<I>orl</I> esta en la posicion:
"+cad.indexOf("orl"));
write("<BR>En cursiva: "+cad.italics()+"<BR>");
write("La primera <I>l</I> esta, empezando a contar por
detras,");
write(" en la posicion: "+cad.lastIndexOf("l")+"<BR>");
write("Haciendola enlace: "+cad.link("doc.htm")+"<BR>");
write("En pequeño: "+cad.small()+"<BR>");
write("Tachada: "+cad.strike()+"<BR>");
write("Subíndice: "+cad.sub()+"<BR>");
write("Superíndice: "+cad.sup()+"<BR>");
write("Minúsculas: "+cad.toLowerCase()+"<BR>");
write("Mayúsculas: "+cad.toUpperCase()+"<BR>");
write("Subcadena entre los caracteres 3 y 10: ");
write(cad.substring(2,10)+"<BR>");
```

```
write("Entre los caracteres 10 y 3:  
"+cad.substring(10,2)+"<BR>");  
write("Subcadenas resultantes de separar por las  
<B>o:</B><BR>");  
for(i=0;i<jc.length;i++) write(jc[i]+"<BR>");  
}  
/-->  
</script>  
</BODY>  
</HTML>
```

El Objeto Array

Este objeto nos va a dar la facilidad de construir arrays cuyos elementos pueden contener cualquier tipo básico, y cuya longitud se modificará de forma dinámica siempre que añadamos un nuevo elemento (y, por tanto, no tendremos que preocuparnos de esa tarea). Para poder tener un objeto array, tendremos que crearlo con su constructor, por ejemplo, si escribimos:

```
a=new Array(15);
```

tendremos creada una variable a que contendrá 15 elementos, enumerados del 0 al 14. Para acceder a cada elemento individual usaremos la notación a[i], donde i variará entre 0 y N-1, siendo N el número de elementos que le pasamos al constructor.

También podemos inicializar el array a la vez que lo declaramos, pasando los valores que queramos directamente al constructor, por ejemplo:

```
a=new Array(21,"cadena",true);
```

que nos muestra, además, que los elementos del array no tienen por qué ser del mismo tipo.

Por tanto: si ponemos un argumento al llamar al constructor, este será el número de elementos del array (y habrá que asignarles valores posteriormente), y si ponemos más de uno, será la forma de inicializar el array con tantos elementos como argumentos reciba el constructor.

Podríamos poner como mención especial de esto lo siguiente. Las inicializaciones que vemos a continuación:

```
a=new Array("cadena");  
a=new Array(false);
```

Inicializan el array a, en el primer caso, con un elemento cuyo contenido es la cadena cadena, y en el segundo caso con un elemento cuyo contenido es false.

Lo comentado anteriormente sobre inicialización de arrays con varios valores, significa que si escribimos

```
a=new Array(2,3);
```

NO vamos a tener un array con 2 filas y 3 columnas, sino un array cuyo primer elemento será el 2 y cuyo segundo elemento será el 3. Entonces, ¿cómo creamos un array bidimensional? (un array bidimensional es una construcción bastante frecuente). Creando un array con las filas deseadas y, después, cada elemento del array se inicializará con un array con las columnas deseadas. Por ejemplo, si queremos crear un array con 4 filas y 7 columnas, bastará escribir:

```
a=new Array(4);  
for(i=0;i<4;i++) a[i]=new Array(7);
```

y para referenciar al elemento que ocupa la posición (i,j), escribiremos a[i][j];

Propiedades

- length. Esta propiedad nos dice en cada momento la longitud del array, es decir, cuántos elementos tiene.
- prototype. Nos permite asignar nuevas propiedades al objeto String.

Métodos

- join(separador). Une los elementos de las cadenas de caracteres de cada elemento de un array en un string, separando cada cadena por el separador especificado.
- reverse(). Invierte el orden de los elementos del array.
- sort(). Ordena los elementos del array siguiendo el orden lexicográfico.

```
<HTML>  
<HEAD>  
  <title>Ejemplo de JavaScript</title>  
</HEAD>  
<BODY>  
<script LANGUAGE="JavaScript">  
  <!--  
    var j=new Array(2),h=new Array(1), i=new  
    Array(1,"Hola",3);  
    var b=new  
    Array("Palabra","Letra","Amor","Color","Cariño");  
    var c=new Array("Otra cadena con palabras");  
    var d=new Array(false);  
  
    j[0]=new Array(3);  
    j[1]=new Array(2);  
  
    j[0][0]=0; j[0][1]=1; j[0][2]=2;  
    j[1][0]=3; j[1][1]=4; j[1][2]=5;  
  
    document.write(c);  
    document.write("<P>" + d + "<P>");  
  
    document.write("j[0][0]=" + j[0][0] + ";
```

```

j[0][1]="+j[0][1]+
"; j[0][2]="+j[0][2]+"<BR>";
document.write("j[1][0]="+j[1][0]+"");
j[1][1]="+j[1][1]+
"; j[1][2]="+j[1][2]);
document.write("<P>h= "+(h[0]='Hola')+"<P>");
document.write("i[0]="+i[0]+"; i[1]="+i[1]+";
i[2]="+i[2]+"<P>");
document.write("Antes de ordenar: "+b.join(',
')+"<P>");
document.write("Ordenados: "+b.sort()+"<P>");
document.write("Ordenados en orden inverso:
"+b.sort().reverse());

//-->
</script>
</BODY>
</HTML>

```

El Objeto Math

Este objeto se utiliza para poder realizar cálculos en nuestros scripts. Tiene la peculiaridad de que sus propiedades no pueden modificarse, sólo consultarse. Estas propiedades son constantes matemáticas de uso frecuente en algunas tareas, por ello es lógico que sólo pueda consultarse su valor pero no modificarlo.

Propiedades

- E. Número 'e', base de los logaritmos naturales (neperianos).
- LN2. Logaritmo neperiano de 2.
- LN10. Logaritmo neperiano de 10.
- LOG2E. Logaritmo en base 2 de e.
- LOG10E. Logaritmo en base 10 de e.
- PI. Número PI.
- SQRT1_2. Raíz cuadrada de 1/2.
- SQRT2. Raíz cuadrada de 2.

Métodos

- abs(numero). Función valor absoluto.
- acos(numero). Función arcocoseno. Devuelve un valor cuyas unidades son radianes o NaN. 'numero' debe pertenecer al rango [-1,1], en otro caso devuelve NaN.
- asin(numero). Función arcoseno. Devuelve un valor cuyas unidades son radianes o NaN. 'numero' debe pertenecer al rango [-1,1], en otro caso devuelve NaN.
- atan(numero). Función arcotangente. Devuelve un valor cuyas unidades son radianes o NaN.
- atan2(x,y). Devuelve el ángulo formado por el vector de coordenadas (x,y) con respecto al eje OX.
- ceil(numero). Devuelve el entero obtenido de redondear 'numero' "por arriba".
- cos(numero). Devuelve el coseno de 'numero' (que debe estar en radianes) o NaN.

- `exp(numero)`. Devuelve el valor `ennumero`.
- `floor(numero)`. Devuelve el entero obtenido de redondear '`numero`' "por abajo".
- `log(numero)`. Devuelve el logaritmo neperiano de '`numero`'.
- `max(x,y)`. Devuelve el máximo de '`x`' e '`y`'.
- `min(x,y)`. Devuelve el mínimo de '`x`' e '`y`'.
- `pow(base,exp)`. Devuelve el valor `baseexp`.
- `random()`. Devuelve un número pseudoaleatorio entre 0 y 1.
- `round(numero)`. Redondea '`numero`' al entero más cercano.
- `sin(numero)`. Devuelve el seno de '`numero`' (que debe estar en radianes) o NaN.
- `sqrt(numero)`. Devuelve la raíz cuadrada de número.
- `tan(numero)`. Devuelve la tangente de '`numero`' (que debe estar en radianes) o NaN.

El Objeto Date

Este objeto nos va a permitir hacer manipulaciones con fechas: poner fechas, consultarlas... para ello, debemos saber lo siguiente: JS maneja fechas en milisegundos. Los meses de Enero a Diciembre vienen dados por un entero cuyo rango varía entre el 0 y el 11 (es decir, el mes 0 es Enero, el mes 1 es Febrero, y así sucesivamente), los días de la semana de Domingo a Sábado vienen dados por un entero cuyo rango varía entre 0 y 6 (el día 0 es el Domingo, el día 1 es el Lunes, ...), los años se ponen tal cual, y las horas se especifican con el formato HH:MM:SS.

Podemos crear un objeto `Date` vacío, o podemos crearlo dándole una fecha concreta. Si no le damos una fecha concreta, se creará con la fecha correspondiente al momento actual en el que se crea. Para crearlo dándole un valor, tenemos estas posibilidades:

```
var Mi_Fecha = new Date(año, mes);
var Mi_Fecha = new Date(año, mes, día);
var Mi_Fecha = new Date(año, mes, día, horas);
var Mi_Fecha = new Date(año, mes, día, horas, minutos);
var Mi_Fecha = new Date(año, mes, día, horas, minutos, segundos);
```

En día pondremos un número del 1 al máximo de días del mes que toque. Todos los valores que tenemos que pasar al constructor son enteros. Pasamos a continuación a estudiar los métodos de este objeto.

Métodos

- `getDate()`. Devuelve el día del mes actual como un entero entre 1 y 31.
- `getDay()`. Devuelve el día de la semana actual como un entero entre 0 y 6.
- `getHours()`. Devuelve la hora del día actual como un entero entre 0 y 23.
- `getMinutes()`. Devuelve los minutos de la hora actual como un entero entre 0 y 59.
- `getMonth()`. Devuelve el mes del año actual como un entero entre 0 y 11.
- `getSeconds()`. Devuelve los segundos del minuto actual como un entero entre 0 y 59.
- `getTime()`. Devuelve el tiempo transcurrido en milisegundos desde el 1 de enero de 1970 hasta el momento actual.
- `getFullYear()`. Devuelve el año actual como un entero.

- setDate(día_mes). Pone el día del mes actual en el objeto Date que estemos usando.
- setDay(día_semana). Pone el día de la semana actual en el objeto Date que estemos usando.
- setHours(horas). Pone la hora del día actual en el objeto Date que estemos usando.
- setMinutes(minutos). Pone los minutos de la hora actual en el objeto Date que estemos usando.
- setMonth(mes). Pone el mes del año actual en el objeto Date que estemos usando.
- setSeconds(segundos). Pone los segundos del minuto actual en el objeto Date que estemos usando.
- setTime(milisegundos). Pone la fecha que dista los milisegundos que le pasemos del 1 de enero de 1970 en el objeto Date que estemos usando.
- setYear(año). Pone el año actual en el objeto Date que estemos usando.
- toGMTString(). Devuelve una cadena que usa las convenciones de [Internet](#) con la zona horaria GMT.

El Objeto Number

Este objeto representa el tipo de dato número con el que JS trabaja. Podemos asignar a una variable un número, o podemos darle valor, mediante el constructor Number, de esta forma:

a = new Number(valor);, por ejemplo, a = new Number(3.2); da a a el valor 3.2. Si no pasamos algún valor al constructor, la variable se inicializará con el valor 0.

Propiedades

- MAX_VALUE. Valor máximo que se puede manejar con un tipo numérico
- MIN_VALUE. Valor mínimo que se puede manejar con un tipo numérico
- NaN. Representación de un dato que no es un número
- NEGATIVE_INFINITY. Representación del valor a partir del cual hay desbordamiento negativo (underflow)
- POSITIVE_INFINITY. Representación del valor a partir del cual hay desbordamiento positivo (overflow)

El objeto window

Se trata del objeto más alto en la jerarquía del navegador (navigator es un objeto independiente de todos en la jerarquía), pues todos los componentes de una página web están situados dentro de una ventana. El objeto window hace referencia a la ventana actual. Veamos a continuación sus propiedades y sus métodos.

Propiedades

- closed. Válida a partir de Netscape 3 en adelante y MSIE 4 en adelante. Es un booleano que nos dice si la ventana está cerrada (closed = true) o no (closed = false).

- `defaultStatus`. Cadena que contiene el texto por defecto que aparece en la barra de estado (status bar) del navegador.
- `frames`. Es un array: cada elemento de este array (`frames[0]`, `frames[1]`, ...) es uno de los frames que contiene la ventana. Su orden se asigna según se definen en el documento HTML.
- `history`. Se trata de un array que representa las URLS visitadas por la ventana (están almacenadas en su historial).
- `length`. Variable que nos indica cuántos frames tiene la ventana actual.
- `location`. Cadena con la URL de la barra de dirección.
- `name`. Contiene el nombre de la ventana, o del frame actual.
- `opener`. Es una referencia al objeto window que lo abrió, si la ventana fue abierta usando el método `open()` que veremos cuando estudiemos los métodos.
- `parent`. Referencia al objeto window que contiene el frameset.
- `self`. Es un nombre alternativo del window actual.
- `status`. String con el mensaje que tiene la barra de estado.
- `top`. Nombre alternativo de la ventana del nivel superior.
- `window`. Igual que `self`: nombre alternativo del objeto window actual.

Métodos

- `alert(mensaje)`. Muestra el mensaje 'mensaje' en un cuadro de diálogo
- `blur()`. Elimina el foco del objeto window actual. A partir de NS 3, IE 4.
- `clearInterval(id)`. Elimina el intervalo referenciado por 'id' (ver el método `setInterval()`, también del objeto window). A partir de NS 4, IE 4.
- `clearTimeout(nombre)`. Cancela el intervalo referenciado por 'nombre' (ver el método `setTimeout()`, también del objeto window).
- `close()`. Cierra el objeto window actual.
- `confirm(mensaje)`. Muestra un cuadro de diálogo con el mensaje 'mensaje' y dos botones, uno de aceptar y otro de cancelar. Devuelve true si se pulsa aceptar y devuelve false si se pulsa cancelar.
- `focus()`. Captura el foco del ratón sobre el objeto window actual. A partir de NS 3, IE 4.
- `moveBy(x,y)`. Mueve el objeto window actual el número de pixels especificados por (x,y). A partir de NS 4.
- `moveTo(x,y)`. Mueve el objeto window actual a las coordenadas (x,y). A partir de NS 4.
- `open(URL,nombre,características)`. Abre la URL que le pasemos como primer parámetro en una ventana de nombre 'nombre'. Si esta ventana no existe, abrirá una ventana nueva en la que mostrará el contenido con las características especificadas. Las características que podemos elegir para la ventana que queramos abrir son las siguientes:
 - `toolbar = [yes|no|1|0]`. Nos dice si la ventana tendrá barra de herramientas (yes,1) o no la tendrá (no,0).
 - `location = [yes|no|1|0]`. Nos dice si la ventana tendrá campo de localización o no.
 - `directories = [yes|no|1|0]`. Nos dice si la nueva ventana tendrá botones de dirección o no.

- status = [yes|no|1|0]. Nos dice si la nueva ventana tendrá barra de estado o no.
- menubar = [yes|no|1|0]. Nos dice si la nueva ventana tendrá barra de menús o no.
- scrollbars = [yes|no|1|0]. Nos dice si la nueva ventana tendrá barras de desplazamiento o no.
- resizable = [yes|no|1|0]. Nos dice si la nueva ventana podrá ser cambiada de tamaño (con el ratón) o no.
- width = px. Nos dice el ancho de la ventana en pixels.
- height = px. Nos dice el alto de la ventana en pixels.
- outerWidth = px. Nos dice el ancho *total* de la ventana en pixels. A partir de NS 4.
- outerHeight = px. Nos dice el alto *total* de la ventana en pixels. A partir de NS 4
- left = px. Nos dice la distancia en pixels desde el lado izquierdo de la pantalla a la que se debe colocar la ventana.
- top = px. Nos dice la distancia en pixels desde el lado superior de la pantalla a la que se debe colocar la ventana.
- prompt(mensaje,respuesta_por_defecto). Muestra un cuadro de diálogo que contiene una caja de texto en la cual podremos escribir una respuesta a lo que nos pregunte en 'mensaje'. El parámetro 'respuesta_por_defecto' es opcional, y mostrará la respuesta por defecto indicada al abrirse el cuadro de diálogo. El método retorna una cadena de caracteres con la respuesta introducida.
- scroll(x,y). Desplaza el objeto window actual a las coordenadas especificadas por (x,y). A partir de NS3, IE4.
- scrollBy(x,y). Desplaza el objeto window actual el número de pixels especificado por (x,y). A partir de NS4.
- scrollTo(x,y). Desplaza el objeto window actual a las coordenadas especificadas por (x,y). A partir de NS4.
- setInterval(expresion,tiempo). Evalua la expresión especificada después de que hayan pasado el número de milisegundos especificados en tiempo. Devuelve un valor que puede ser usado como identificador por clearInterval(). A partir de NS4, IE4.
- setTimeout(expresion,tiempo). Evalua la expresión especificada después de que hayan pasado el número de milisegundos especificados en tiempo. Devuelve un valor que puede ser usado como identificador por clearTimeout(). A partir de NS4, IE4.

Existen otras propiedades y métodos como innerHeight, innerWidth, outerHeight, outerWidth, pageXOffset, pageYOffset, personalbar, scrollbars, back(), find(["cadena"],[caso,bkwd]), forward(), home(), print(), stop()... todas ellas disponibles a partir de NS 4 y cuya explicación remito como ejercicio al lector interesado en saber más sobre el objeto window.

```
<HTML>
<HEAD>
  <title>Ejemplo de JavaScript</title>
  <script LANGUAGE="JavaScript">
```

```

<!--
function moverVentana()
{
    mi_ventana.moveBy(5,5);
    i++;
    if (i<20)
        setTimeout('moverVentana()',100);
    else
        mi_ventana.close();
}
//-->
</script>

</HEAD>
<BODY>
<script LANGUAGE="JavaScript">
<!--
    var
    opciones="left=100,top=100,width=250,height=150", i=
    0;

    mi_ventana = window.open("", "", opciones);
    mi_ventana.document.write("Una prueba de abrir
    ventanas");
    mi_ventana.moveTo(400,100);
    moverVentana();
    //-->
</script>
</BODY>
</HTML>

```

El objeto document

El objeto document es el que tiene el contenido de toda la página que se está visualizando. Esto incluye el texto, imágenes, enlaces, formularios, Gracias a este objeto es posible añadir dinámicamente contenido a la página, o hacer cambios, según convenga.

Propiedades

- **alinkColor.** Esta propiedad tiene almacenado el color de los enlaces activos
- **anchors.** Se trata de un array con los enlaces internos existentes en el documento
- **applets.** Es un array con los applets existentes en el documento
- **bgColor.** Propiedad que almacena el color de fondo del documento
- **cookie.** Es una cadena con los valores de las cookies del documento actual
- **domain.** Guarda el nombre del [servidor](#) que ha servido el documento
- **embeds.** Es un array con todos los EMBED del documento
- **fgColor.** En esta propiedad tenemos el color del primer plano
- **forms.** Se trata de un array con todos los formularios del documento. Los formularios tienen a su vez elementos (cajas de texto, botones, etc) que tienen sus propias propiedades y métodos, y serán tratados en el siguiente capítulo.
- **images.** Array con todas las imágenes del documento
- **lastModified.** Es una cadena con la fecha de la última modificación del documento

- linkColor. Propiedad que almacena el color de los enlaces
- links. Es un array con los enlaces externos
- location. Cadena con la URL del documento actual
- referrer. Cadena con la URL del documento que llamó al actual, en caso de usar un enlace.
- title. Cadena con el título del documento actual
- vlinkColor. Propiedad en la que se guarda el color de los enlaces visitados

Métodos

- clear(). Limpia la ventana del documento
- open(). Abre la escritura sobre un documento.
- close(). Cierra la escritura sobre el documento actual
- write(). Escribe texto en el documento.
- writeln(). Escribe texto en el documento, y además lo finaliza con un salto de línea

El objeto form

Este objeto es el contenedor de todos los elementos del formulario. Como ya vimos al tratar el objeto document, los formularios se agrupan en un array dentro de document. Cada elemento de este array es un objeto de tipo form.

Propiedades

- action. Es una cadena que contiene la URL del parámetro ACTION del form, es decir, la dirección en la que los datos del formulario serán procesados.
- elements. Es un array que contiene todos los elementos del formulario, en el mismo orden en el que se definen en el documento HTML. Por ejemplo, si en el formulario hemos puesto, en este orden, una caja de texto, un checkbox y una lista de selección, la caja de texto será elements[0], el checkbox será elements[1] y la lista de selección será elements[2].
- encoding. Es una cadena que tiene la codificación mime especificada en el parámetro ENCTYPE del form.
- method. Es una cadena que tiene el nombre del método con el que se va a recibir/procesar la información del formulario (GET/POST).

Métodos

- reset(). Resetea el formulario: tiene el mismo efecto que si pulsáramos un botón de tipo RESET dispuesto en el form.
- submit(). Envía el formulario: tiene el mismo efecto que si pulsáramos un botón de tipo SUBMIT dispuesto en el form.

Los objetos text, textarea y password

Estos objetos representan los campos de texto dentro de un formulario. Además, el objeto password es exactamente igual que el text salvo en que no muestra los caracteres introducidos por el usuario, poniendo asteriscos (*) en su lugar.

Propiedades

- `defaultValue`. Es una cadena que contiene el valor por defecto que se le ha dado a uno de estos objetos por defecto.
- `name`. Es una cadena que contiene el valor del parámetro NAME.
- `value`. Es una cadena que contiene el valor del parámetro VALUE.
- `maxLength`. Número máximo de caracteres que puede contener el campo de texto.

Métodos

- `blur()`. Pierde el foco del ratón sobre el objeto especificado.
- `focus()`. Obtiene el foco del ratón sobre el objeto especificado.
- `select()`. Selecciona el texto dentro del objeto dado.

Nota: El material que se incluye en esta sección fue tomado en su mayoría de <http://www.webestilo.com/javascript/>

1.4 Estándares de calidad

El World Wide Web Consortium, abreviado W3C, es un consorcio internacional que produce estándares para la World Wide Web. Está dirigida por Tim Berners-Lee, el creador original de URL (Uniform Resource Locator, Localizador Uniforme de Recursos), HTTP (HyperText Transfer Protocol, Protocolo de Transferencia de HiperTexto) y HTML (Lenguaje de Marcado de HiperTexto) que son las principales tecnologías sobre las que se basa la Web.

Los estándares existen y desde siempre han sido pieza fundamental para mantener el avance tecnológico hacia una misma dirección, asegurando la compatibilidad entre los usuarios de diferentes partes del mundo.

Sin embargo, enfocándose específicamente a la Web ¿Realmente influye el respeto a los estándares web en el posicionamiento de una página? ¿Supone la validación una garantía de éxito o, por el contrario, que tu sitio no valide afecta a su presencia en la Red? Mucho se ha escrito sobre el tema y muchos son los que afirman que sí, que sin una correcta adecuación de un código a los preceptos de la W3C, la web poco tiene que hacer en esta enorme maraña de propuestas que es Internet.

Sin embargo, la práctica parece contradecir esta supuesta ‘verdad teórica’: a través de un sencillo experimento se puede comprobar que, de los primeros 13 sitios con mayor éxito en la Red, según Alexa (www.alexa.com), tan solo dos de ellos pasan el test de validación. ¿Cuáles? La Wikipedia y MSN.

La gran mayoría de los sitios que aparecen en las primeras posiciones de su ránking son ampliamente reconocidos como los más populares de la Red, independientemente de que pueda haber otros o de que estos mismos puedan estar realmente uno o dos puestos por encima o por debajo. Son, en cualquier caso, una excelente muestra de lo ‘más de lo más’ en Internet. A continuación no hay más que acudir a la herramienta de validación de la W3C y pasarle el test a sus páginas principales. Éste es el resultado, organizado según el ranking de Alexa:

1. [Yahoo!](#): no valida - [33 errores](#)
2. **[MSN](#)**: valida - **0 errores**
3. [Google](#): no valida - [68 errores](#)
4. [YouTube](#): no valida - [181 errores](#)
5. [MySpace](#): no valida - [289 errores](#)
6. [Windows Live](#): no valida - [60 errores](#)
7. [Baidu](#): no valida - [30 errores](#)
8. [Orkut](#): no valida - [28 errores](#)
9. [qq](#): no valida - [477 errores](#)
10. [Yahoo! Japón](#): no valida - [398 errores](#)
11. **[Wikipedia](#)**: valida - **0 errores**
12. [Sina](#): no valida - [19 errores](#)
13. [Microsoft](#): no valida - [5 errores](#)

¿Quiere decir esto que mantener tu sitio fiel a los estándares no sirve para nada? En absoluto. La relación entre validación y posicionamiento es sólo una de las ventajas, bastante relativa, por cierto, como hemos podido comprobar, de los estándares. Lo que sí quiere decir, es que ni validar es garantía de éxito en el posicionamiento, ni los errores de validación son sinónimo de fracaso. Sobre todo si tenemos en cuenta que, curiosamente, ¡ninguno de los buscadores analizados, que en teoría tienen en cuenta los estándares a la hora de otorgar ‘importancia’ a las páginas web, pasan el test de validación! Es decir: Google, Windows Live y Yahoo!

Probablemente todo tenga que ver con una ponderación relativa en la que los estándares ocupen una determinada cuota de influencia junto a otros factores de peso como utilidad, calidad de contenidos y otras estrategias.

En todo caso, la recomendación sigue siendo la misma: intenta mantener tu blog o web lo más cercana a los estándares posibles. Quizás el sitio desarrollado no aparezcas nunca en el Top 500 de Alexa o en cualquier otro ranking de ‘lo mejor de la web’, pero tu prestigio, tu servidor, tu tiempo y, sobre todo, tus usuarios te lo agradecerán.

A propósito del párrafo anterior, algo casi imprescindible hoy en día para mejorar el posicionamiento en buscadores consiste en transformar nuestros viejos documentos HTML

a XHTML. Para los nuevos sitios es casi una obligación empezar utilizando XHTML, ya que aseguraremos una mayor compatibilidad con todos los navegadores y mejoraremos las posibilidades de aparecer en las posiciones más altas de los buscadores.

¿Qué es el XHTML? XHTML, al igual que HTML, es un estándar propuesto por el W3C y basado en XML. Desde que nació el HTML ha habido una invención constante de nuevos elementos (etiquetas o atributos) para ser usados dentro de HTML, lo que ha provocado grandes problemas de compatibilidad entre las distintas plataformas. El XHTML es una "reformulación del estándar HTML" que pretende conseguir que todos los documentos web sean compatibles en cualquier navegador de cualquier dispositivo.

Pero... ¿por qué XHTML ayuda al posicionamiento web? Utilizando XHTML se consigue que el código de nuestras páginas web sean mucho más sencillo, limpio y claro para los robots de los buscadores.

Si además se utilizan hojas de estilo CSS para todo lo referente a la presentación, el código de nuestras páginas será todavía mucho más limpio, tendrá bastantes menos etiquetas y atributos con lo que aumentará el porcentaje de palabras clave y se reducirá el peso de las páginas y el tiempo de carga.

Características del XHTML

- Todos los **nombres de etiquetas** y sus atributos deben estar escritos en minúsculas.
- Los **valores de los atributos** deben ir siempre entre comillas.
- Todas las **etiquetas de apertura y cierre** tienen que estar anidadas correctamente.
- Todas las **etiquetas** deben tener su correspondiente etiqueta de cierre. Los elementos que estén vacíos deben tener también etiqueta de cierre o terminar con />
- Atributos como **selected** o **checked** tienen que escribirse de forma completa: selected="selected" y checked="checked" (Ya no se permiten este tipo de abreviaturas).
- Los documentos XHTML deben incluir obligatoriamente una **declaración de tipo de documento** justo antes del nodo raíz.
- El elemento raíz será obligatoriamente y tendremos que declarar también obligatoriamente el **namespace** usando el atributo xmlns:

Y ahora que ya lo he modificado todo, ¿cómo compruebo que está bien?
Una vez que hayas hecho los cambios pertinentes en el código de las páginas, puedes **validar** el código en la siguiente URL que nos brinda el W3C: <http://validator.w3.org/>

Si un sitio cumple con los estándares, puede añadir un icono para indicar a los visitantes que ese sitio cumple con el estándar XHTML 1.0 del W3C.

Nota: El material de esta sección fue tomado de <http://estandaresyusabilidad.blogspot.com/> y <http://mangasverdes.es/>

1.5 Estilos CSS

Con la idea tradicional de elaboración de páginas web, el desarrollador realiza la página web y posteriormente añade las etiquetas para definir la apariencia de la misma. Sin embargo, al utilizar los parámetros para las etiquetas resulta tedioso modificar la apariencia de algún elemento pues es necesario actualizar todas las páginas que integran el sitio. Con el uso de estilos desaparece este problema, como se explica a continuación.

La idea de CSS es separar el contenido de las páginas de su apariencia. Idealmente, las propiedades que se desea aplicar a un "texto grande, centrado y colorido" se escribe en un archivo cuya extensión es .css. En ese archivo se definen todos los estilos que se desean aplicar en las páginas web.

Todos los archivos html harán referencia a ese fichero style.css. Los párrafos que se desean "grandes, centrados y coloridos" simplemente indicaran que tienen el estilo "parrafo importante".

```
<p class="parrafo importante">Este texto es importante</p>
```

Y así se incluye en todas las páginas html del sitio.

Cuando eventualmente se desee cambiar "texto grande, centrado y colorido" por "texto extra grande, cursivo de fondo amarillo chillón", sólo será necesario modificar el archivo style.css, sin cambiarle el nombre "parrafo importante". De esta manera, se cambiarán todos los "parrafo importante" de las páginas web del sitio.

La sintaxis empleada para definir los estilos es la siguiente:

```
selector {  
  propiedad:valor;  
  propiedad:valor;  
  ...  
}
```

Donde:

- `selector` es una etiqueta de html o un nombre dado a un estilo concreto. Si se incluye una etiqueta de html, por ejemplo `p`, el estilo afectará a todos los párrafos de la página html. También se pueden hacer combinaciones espaciales de etiquetas y nombres.
- `propiedad` es el nombre de alguna propiedad que se pueda cambiar, como fuente de letra, tamaño de la fuente, color, etc, etc.

- `valor` es el valor que se desea asignar a esa propiedad. Por ejemplo, si la propiedad es el color de fondo, podemos poner cosas como rojo, verde, etc.

Un ejemplo concreto

```
p {  
  color: red;  
}
```

Esto hará que nuestros párrafos en todo el texto html salgan en rojo.

Dónde colocar los estilos

Existen tres posibilidades para incluir estilos a una página web:

- Directamente en las etiquetas html
- Dentro del archivo html, en la cabecera de la página.
- En un archivo .css separado

CSS en los tag de html

Es posible incluir el estilo de CSS para una etiqueta html directamente, las propiedades y valores se indican entre comillas y haciendo `style="..."`.

```
<p style="color:red;">Esto sale rojo</p>
```

De esta manera se pierden las ventajas de CSS pues si más adelante se requiere modificar el estilo, será necesario modificar todos los archivos html uno a uno y en cada uno de ellos todas las etiquetas afectadas, que pueden ser muchas.

Esta opción se reserva sólo para casos muy particulares, para algún párrafo o etiqueta html muy específico que sólo requiere ese estilo. Por ejemplo, un párrafo con formato de bienvenida a un sitio y que no se va a usar más en ninguna otra página del mismo.

Entre los tags head de la página html

Otra opción es poner el estilo entro las etiquetas <head>...</head> de la página html:

```
<html>
  <head>
    ...
    <style type="text/css">
      p {
        color:red;
      }
    </style>
    ...
  </head>
  ...
```

De esta manera, el estilo afectará a todos los párrafos de la página html.

Esta opción es algo mejor que la anterior, pero será necesario repetir esto en todas las páginas html que tengan los mismos estilos. Si se requiere modificar algo, será necesario modificar todas las páginas html. Suele reservarse esta forma para estilos que se repiten en varios sitios de la misma página html, pero que no se piensa usar más en las otras páginas html del sitio.

En fichero .css separado

Esta es posiblemente la mejor opción. Se incluyen todos los estilos CSS en un archivo separado que habitualmente tiene la extensión .css. Un ejemplo de estilo.css puede ser:

```
p {
  color:red;
}
```

Luego, en el archivo html, entre las etiquetas <head> se incluye:

```
<html>
  <head>
    <link rel="stylesheet" href="estilo.css"
    type="text/css" />
    ...
  </head>
  ...
```

De esta forma el estilo del archivo estilo.css se aplicará a la página html y todos los párrafos saldrán en color rojo.

Los demás archivos html del sitio pueden incluir de la misma forma el mismo estilo.css y así saldrán todos los párrafos de todas las páginas del sitio de color rojo. Si un día ya no se desea usar más el rojo sino el azul, solamente habría que cambiar "red" por "blue" en el fichero estilo.css y nos saldrá todo en azul.

Un ejemplo más completo y en el que se aplica un estilo sólo a un determinado tipo de párrafos podría ser el siguiente:

```
estilo.css
/* Indicamos que los parrafos importantes deben ser
rojos */
p,importante {color:red};
Mientras que el fichero html puede contener esto
fichero.html
<html>
  <head>
    <link rel="stylesheet" type="text/css"
href="./estilo.css">
  </head>
  <body>
    <p>Esto saldra normal</p>
    <p class="importante">Esto saldra rojo</p>
  </body>
</html>
```

Lo de "importante" es un nombre que hemos elegido nosotros para identificar los párrafos importantes y su estilo (color rojo).

Para integrar una página web con **CSS**, es necesario definir las secciones principales de las páginas que integrarán el sitio. Por ejemplo: un título, un texto largo, unos anuncios publicitarios y una lista de enlaces de interés.

Entonces, la página html deberá contener estos cuatro bloques, usando **<div>** y un **id** para cada uno de ellos. Puede ser esto

```
<html>
<head>
  <link rel="stylesheet" type="text/css"
href="./estilo.css">
  <title>Ejemplo de bloques css</title>
</head>
<body>
  <div id="enlaces">enlaces</div>
  <div id="titulo">TITULO</div>
  <div id="texto">aqui el texto</div>
  <div id="anuncio">publicidad</div>
</body>
</html>
```

Propiedades de fuente

Propiedad Valor

font-family

Fuente específica (Arial, Times, Verdana)
Familia (serif, sans-serif,

Descripción

Define uno o más nombres de fuentes o familias de fuentes. Si se definen múltiples fuentes, se utilizará la primera que se encuentre en el sistema del usuario.

	fantasy, monospace, cursive)	
font-style	normal, italic, oblique	Define el estilo de la escritura
font-weight	lighter, normal, bold o bolder. Valor numérico (100, 200, 300, 400, 500, 600, 700, 800, 900)	Define el grosor de la fuente
font-size	xx-small, x-small, small, medium, large, x-large, xx-large Tamaño en puntos (pt), cm, %	Define el tamaño de la fuente
font-variant	normal, small-caps	Define una variante (mayúsculas chicas)
font	font: Verdana, Arial, bold italic 8px;	Acceso directo a todas las propiedades

Textos y párrafos

Propiedad	Valor	Descripción
color	"#RRGGBB"	Define el color del texto
line-height	line-height: 12pt;	Define el espacio entre las líneas
text-align	left, center, right o justify	Define la alineación del texto
text-indent	text-indent: 5px;	Define la sangría
text-decoration	<i>blink</i> (parpadeo), <i>underline</i> (subrayado), <i>line-through</i> (tachado), <i>overline</i> (línea sobre el texto) o <i>none</i> (sin decoración)	Define la decoración
text-shadow	text-shadow: 1px 2px 4px black;	Define una sombra paralela del texto y representa, respectivamente, la sombra hacia la derecha, hacia abajo, radio de desenfoque y color.
text-transform	<i>uppercase</i> (mayúscula), <i>lowercase</i> (minúscula) o <i>capitalize</i> (primea letra en mayúscula)	Define la capitalización del texto
white-space	normal (el texto continuará en la próxima línea), pre (el texto aparecerá con los espacios en blanco que se ingresaron), nowrap (el texto no continuará)	División de palabras
word-spacing	word-spacing: 6px;	Define cuánto espacio insertar entre las palabras
width	en puntos (pts), pulgadas ("),	Define el ancho de un texto o una

	centímetros, píxeles (px) o en %	imagen
height	en puntos (pts), pulgadas ("), centímetros, píxeles (px) o en %	Define la altura de un texto o una imagen

Colores de fondo

Propiedad	Valor	Descripción
background-color	"#RRGGBB"	Define el color de fondo
background-image	url(http://url)	Define la imagen de fondo
background-repeat	repeat, repeat-x, repeat-y, no-repeat	Define cómo se repite la imagen de fondo
background-attachment	scroll, fixed	Especifica si la imagen de fondo se quedará en su lugar cuando la pantalla se desplace
background-position	top, middle, bottom, left, center o right	Posiciona la imagen con respecto a la esquina superior izquierda
background	background: url(test.jpg) fixed repeat;	Acceso directo a las propiedades de fondo

Márgenes

Propiedad	Ejemplo	Descripción
margin-top	margin-top: 5px;	Valor del margen superior
margin-right	margin-right: 0.5em;	Valor del margen derecho
margin-bottom	margin-bottom: 2pt;	Valor del margen inferior
margin-left	margin-left: 0;	Valor del margen izquierdo
margin	margin: 5px 0.5em 2pt 0;	Acceso directo a las propiedades de márgenes

Bordes

Propiedad	Valor	Descripción
border[-top -left -bottom -right]-width	en puntos (pts), pulgadas ("), centímetros, píxeles (px) o en %	Grosor del borde (para la ubicación dada)
border[-top -left -bottom -right]-color	border-left-color: #RRGGBB;	Color del borde (para la ubicación dada)
border[-top -left -bottom -right]-style	<i>solid</i> (sólido), <i>dashed</i> (con trazos), <i>dotted</i> (con puntos), <i>double</i> (dos líneas) o <i>ridge</i> (tridimensional)	Estilo del borde (para la ubicación dada)
border-collapse	collapse separate	Agrega o elimina el efecto "3D"
Border	border: 1px 0 0 2px dotted green;	Acceso directo global a las propiedades de bordes

Relleno

Propiedad	Valor	Descripción
padding-top	padding-top: 3px;	Relleno entre el elemento y el borde superior
padding-right	padding-right: 0.25em;	Relleno entre el elemento y el borde derecho
padding-bottom	padding-bottom: 0;	Relleno entre el elemento y el borde inferior
padding-left	padding-left: 2pt;	Relleno entre el elemento y el borde izquierdo
padding	padding: 3px 0.25em 0 2pt;	Acceso directo a las propiedades de relleno

Tablas

Propiedad	Valor	Descripción
border-collapse	<i>separate</i> o <i>collapse</i>	Combina los bordes de las celdas (<i>collapse</i>), no los combina (<i>separate</i>)
border-spacing	border-spacing: 4px;	Espacio de las celdas
caption-side	top, bottom, left o right	Ubica la leyenda de la tabla
empty-cells	<i>show</i> o <i>collapse</i>	Muestra (<i>show</i>) u oculta (<i>collapse</i>) las celdas vacías
table-layout	<i>fixed</i> (independiente del contenido de la celda) o <i>auto</i> (depende del contenido de la celda)	Ancho fijo o variable
speak-headers	<i>always</i> (siempre antes de cada celda) o <i>once</i> (sólo una vez)	Propiedad destinada para los ciegos y minusválidos visuales que indica cómo actúa el sonido al leer las celdas de encabezado de las tablas

Listas

Propiedad	Valor	Descripción
list-style-type	decimal, upper-roman, lower-latin, disc, circle, square o none	Tipo de viñetas y numeración
list-style-image	list-style-image: url(image.png);	Personaliza las viñetas con una imagen
list-style-position	inside o outside	Especifica la sangría de las viñetas
list-style		Acceso directo a las propiedades de lista

Presentación de la página

Propiedad	Valor	Descripción
@page	@page(size: portrait)	Define la presentación de impresión
size	auto, landscape o portrait	Formato de impresión

margin-top	margin-top: 3 cm;	Margen superior
margin-right	margin-right: 1,5 cm;	Margen derecho
margin-bottom	margin-bottom: 1 cm;	Margen inferior
margin-left	margin-left: 2 cm;	Margen izquierdo
marks	crop (marcas de recorte), cross (marcas cruzadas), none (sin marcas)	Marcas de recorte y marcas cruzadas
page-break-before	Always, avoid	Inserta un salto de página antes de un elemento
page-break-after	Always, avoid	Inserta un salto de página después de un elemento
orphans	orphans: 2;	Evita que haya líneas huérfanas al final de una página. Define la cantidad mínima de líneas de un elemento que quedan en la parte inferior de una página antes del salto de página.
widows	widows: 1;	Evita que haya líneas viudas al final de una página. Define la cantidad mínima de líneas de un elemento que quedan en la parte superior de una página después de un salto de página.

Unidad 2. Servidores Web, PHP y MySQL

Las páginas estáticas permiten la publicación de contenidos fijos en la Web, sin embargo, actualmente la gran mayoría de las empresas desea añadir dinamismo a sus sitios con el objetivo de estar más cerca de sus clientes, con un contacto directo entre los clientes y la empresa. Una de las formas más poderosas de añadir dinamismo a los sitios web es mediante las páginas dinámicas creadas con PHP, las cuales al conectarse con un servidor de bases de datos permiten todo tipo de transacciones en línea.

En esta unidad se describen las características y pasos para instalar los servidores necesarios para la puesta en marcha de un servidor web con manejo de páginas dinámicas a través de php y mysql.

2.1 *Características de un servidor Web*

Un servidor Web es un programa que sirve datos en forma de páginas Web, hipertextos o páginas HTML (HyperText Markup Language): textos complejos con enlaces, figuras, formularios, botones y objetos incrustados como animaciones o reproductores de sonidos.

La comunicación de estos datos entre cliente y servidor se hace por medio un protocolo, concretamente del protocolo HTTP. Con esto, un servidor Web se mantiene a la espera de peticiones HTTP, que son ejecutadas por un cliente HTTP; lo que solemos conocer como un navegador Web.

A modo de ejemplo: al teclear `http://www.cnice.mec.es` en un navegador, éste realizará una petición HTTP al servidor que tiene asociada dicha URL. El servidor responde al cliente enviando el código HTML de la página; el navegador cuando recibe el código, lo interpreta y lo muestra en pantalla.

El cliente es el encargado de interpretar el código HTML, es decir, de mostrar las fuentes, los colores y la disposición de los textos y objetos de la página. El servidor se encarga de transferir el código de la página sin llevar a cabo ninguna interpretación de la misma.

Los servidores de Hosting permiten la administración, despliegue y servicio de alojamiento en la Web, es decir, proporcionan un espacio en la Red para poder alojar las páginas creadas para cierta empresa o institución. Por otro lado, los dominios son aquellos nombres a través de los cuales se puede acceder a un sitio Web, tales como `www.uady.mx`, `www.yucatan.gob.mx`, etc, lo cual se consigue gracias a un redireccionamiento que al escribir en el navegador la dirección del dominio, internamente se hace una traducción hasta encontrar la dirección IP a la cual pertenece dicho nombre de dominio.

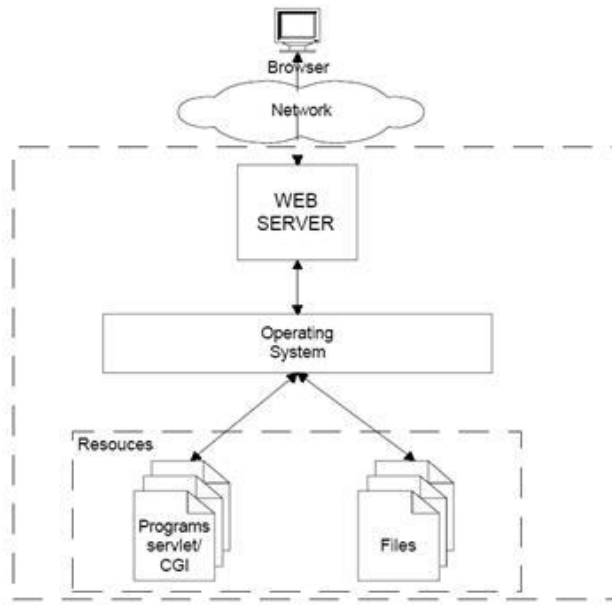


Figura 1. Esquema funcional de un Servidor Web

Son dos conceptos diferentes, pero es muy común que se preste a confusión el significado de cada uno de ellos, por lo que se intenta aclarar la principal diferencia entre ambos: los hosting proporcionan espacio en disco duro y los dominios proporcionan una dirección de acceso con letras en vez de números (la dirección ip es más difícil de memorizar que el nombre de dominio).

Existen hosting gratuitos y comerciales, en el curso se emplean los del primer tipo pero se recomienda emplear el segundo en aplicaciones comerciales por el soporte y seguridad necesarios en un sistema formal.

Ahora bien, los hosting requieren que en la máquina servidor exista un software de administración, al cual se le conoce como Servidor Web. En la siguiente sección se describen las principales características que debe poseer un servidor Web.

En cuanto al hosting, es importante considerar:

- Espacio permitido de almacenamiento en el disco duro del servidor
- Tasa de transferencia máxima permitida periódicamente
- Programas que pueden emplearse en las páginas Web (php, asp, javascript, etc.)
- Capacidad de manejo de bases de datos y correo electrónico
- Soporte (en el caso de los comerciales)

En cuanto a la capacidad de almacenamiento y procesamiento, los requerimientos son cada vez más exigentes, así que se sugiere trabajar con tecnología reciente, aunque no necesariamente de punta pues los costos también son importantes.

Sin embargo, existen ciertos aspectos que es importante tomar en cuenta al decidir emplear algún servidor web, tales como:

1. Facilitar las tareas de administración, despliegue y servicio.

2. Flexibilidad y rendimiento en un servidor.
3. Seguridad en las transacciones realizadas.

Actualmente, existen servidores y hosting gratuitos. Uno de los servidores Web gratuitos más populares desde hace varios años es el Apache, cuya instalación y configuración se describe en la siguiente sección.

2.2 Instalación y configuración de un Servidor Web Apache

Lo primero que hay que hacer es instalar el servidor web; en este caso se describe la instalación de apache en su versión para Windows.

Apache es un servidor open source y el más usado actualmente, se puede encontrar toda la información sobre Apache en su página web: <http://www.apache.org/>, donde también se encuentra, el archivo de instalación: `apache_2.0.43-win32-x86-no_ssl.exe` (puede variar dependiendo de la versión existente).

El proceso de descarga se describe con mayor detalle en el Paquete didáctico de la misma asignatura, creada por los mismos autores. Una vez descargado el archivo de instalación (7 MB aprox.) es necesario instalarlo, para lo cual se pueden seguir los siguientes pasos:

La instalación inicia con la siguiente ventana:



Se requiere hacer clic en “Next” para llegar a la pantalla que muestra la siguiente figura.



Una vez leída la licencia del y haber selecciona la casilla de aceptación (I accept the terms in the licence agreement), hacemos clic en el botón “next”, apareciendo la siguiente ventana:



Al hacer clic en el botón next, aparece otra ventana:



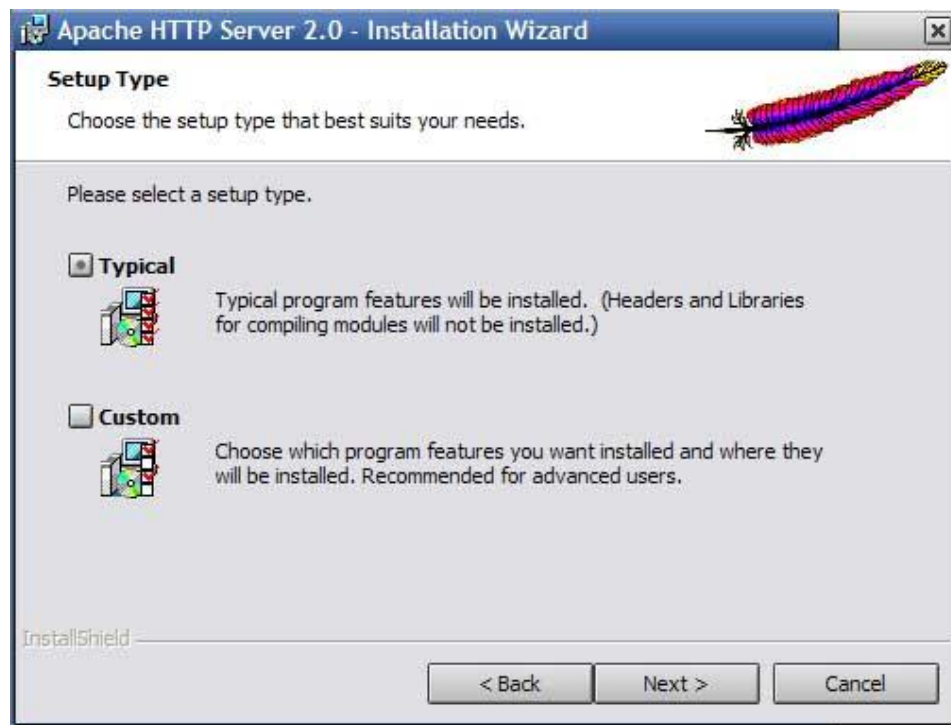
Los datos que se requiere indicar son:

- Network Domain: - En esta casilla tenemos varias opciones; lo más normal es instalar el servidor para usarlo nosotros solos, si este es tu caso en esta casilla pon "localhost" (sin las comillas). De lo contrario, si se desea que el resto del mundo mundial pueda ver las maravillas que puedes hacer con PHP+MySQL pues en esta casilla debes poner tu dirección IP. Nota: Si no sabes cual es tu dirección IP lo puedes averiguar ejecutando, en modo consola, el comando ipconfig.
- Server Name: Es el nombre que quieres que tenga tu servidor web, por ejemplo My_Server.
- Administrator's Email adress : aquí tienes que poner la dirección de correo electrónico del administrador del servidor web; vamos la tuya, por ejemplo: yo_mimo@hotmail.com

Después de haber rellenado estas casillas aparecen dos opciones:

- For all users in port 80, as a service: instala Apache como un servicio de Windows, es decir que Apache se ejecuta al iniciar el ordenador; eligiendo esta opción el servidor se pone a la escucha en el puerto 80.
- Only for the current user , on port 8080, when started manually: instala Apache como un programa normal, para ejecutar el servidor lo elegimos en el menú de inicio y se abre una ventana para indicar que se está ejecutando Apache.

Una vez rellenados todos los campos y elegida la opción que deseemos, hacemos clic en el botón next; nos aparecerá esta ventana:



Esta es la ventana en la escogemos si queremos hacer una instalación típica o personalizada, hacemos clic en la instalación típica (typical) y luego en el botón next.



Aquí se elige el directorio donde se desea instalar el Apache, por defecto se instala en Archivos de programa, pero es posible instalar en c:\Apache\ o en algún otro directorio.

Para cambiar el directorio de instalación hacemos clic en el botón change y escribimos c:\Apache; aceptamos clic en el botón ok y luego en el botón next.

Después de esto nos saldrá una ventanita de confirmacion y tal, hacemos clic en el botón next y comenzará la instalación de Apache.

Cuando acabe la instalación aparecerá otra ventana, simplemente hacemos clic en el boton finish y ya habrá acabado la instalación de nuestro servidor.

Configuración del Servidor Apache.

Una vez instalado, se requiere configurar el servidor. Primero es necesario ejecutar el servidor apache eligiendo la opción start Apache in console, abriéndose una ventana (negra) indicando que se está ejecutando el servidor Apache. Después de esto abrimos nuestro explorador de Internet para ver si realmente nuestro servidor está funcionando. Una vez abierto, tenemos varias opciones según como hayamos rellenado el campo network domain:

- Si hemos puesto nuestra dirección IP, en la barra de dirección de nuestro explorador ponemos esto: http://nuestra direccion ip:8080

Nota: ponemos al final :8080 para indicar al explorador que nuestro servidor está escuchando en el puerto 8080.

- Si en el campo network domain hemos puesto localhost, para que nuestro servidor no sea visible desde Internet pondremos esto en la barra de dirección de nuestro navegador: http://localhost:8080

Si todo es correcto aparecerá una página diciendo que nuestro servidor Apache está configurado con éxito.

Bien, lo siguiente es modificar la configuración de Apache a nuestro gusto. Toda la información del servidor se guarda en el fichero de texto c:\Apache\Apache2\conf\httpd.conf , vamos a abrir el fichero y cambiar algo:

Abrimos el archivo y buscamos el siguiente texto:

```
#
# DocumentRoot: The directory out of which you will serve your
# documents. By default, all requests are taken from this directory, but
# symbolic links and aliases may be used to point to other locations.
#
DocumentRoot "C:/Apache/Apache2/htdocs"
```

Esta es la carpeta donde vamos a tener los archivos .php , .html , etc lo mejor es cambiarla, por ejemplo vamos a poner:

DocumentRoot "C:/servidor_web"

Nota: es importantes fijarse que la barra es esta "/" no esta "\"

El siguiente texto a buscar es este:

```
#  
# This should be changed to whatever you set DocumentRoot to.  
#  
Directory "C:/Apache/Apache2/htdocs"
```

Aqui tenemos que hacer lo mismo antes, sustituir el directorio por el que deseemos, por ejemplo:

Directory "C:/servidor_web"

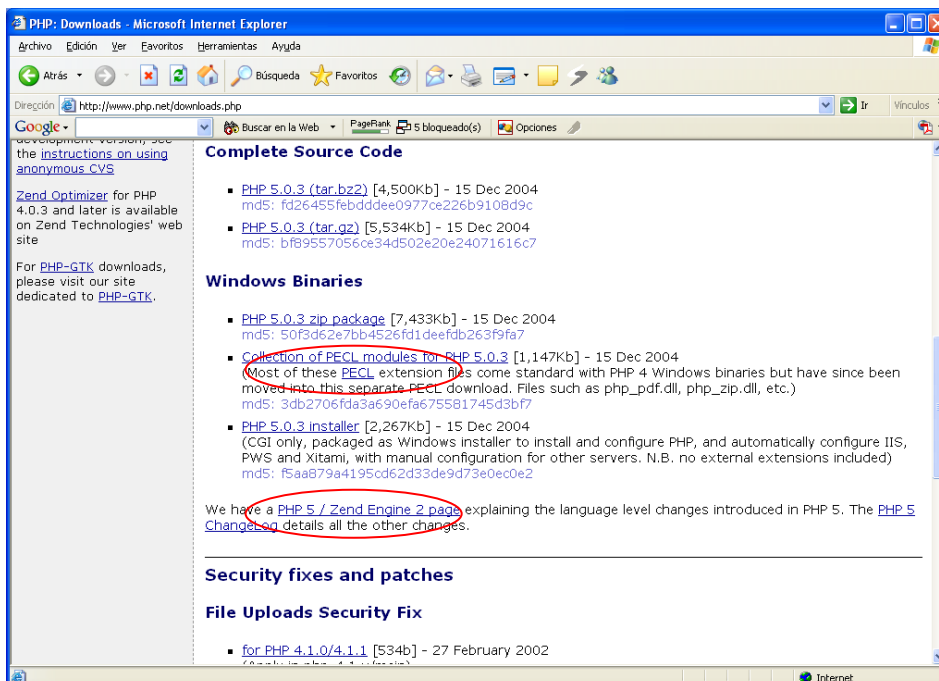
Una vez hecho lo anterior, termina la instalación y configuración de nuestro servidor Apache, de forma básica, simplemente para publicar páginas web, para que luego podamos trabajar con PHP, vamos a tener que hacer modificaciones, las cuales se explican en la siguiente sección.

2.3 Instalación y configuración del servidor PHP

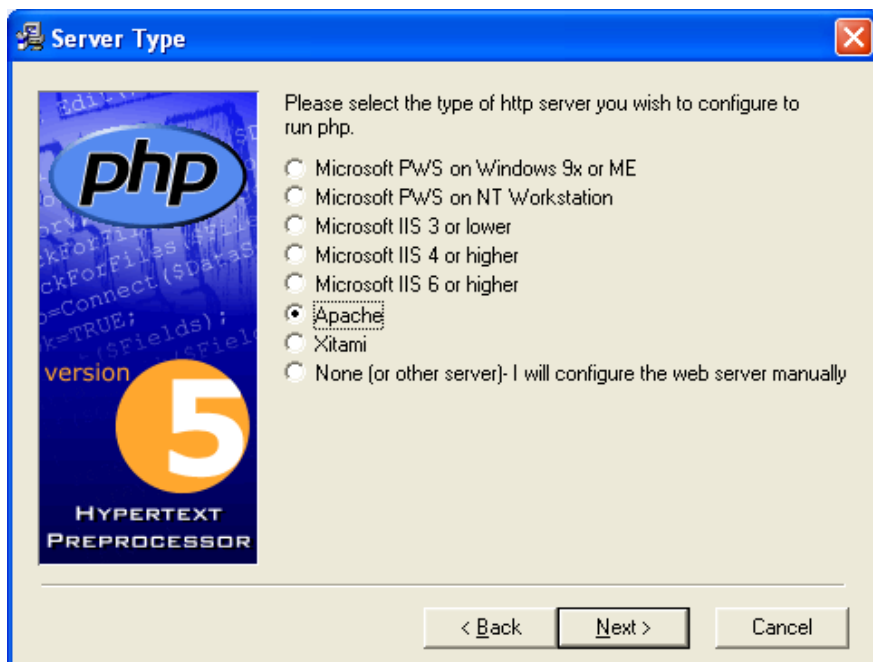
Una vez que terminamos de instalar el servidor WEB ahora vamos a instalar el PHP por lo que en la pagina <http://www.php.net> pulsamos donde dice downloads



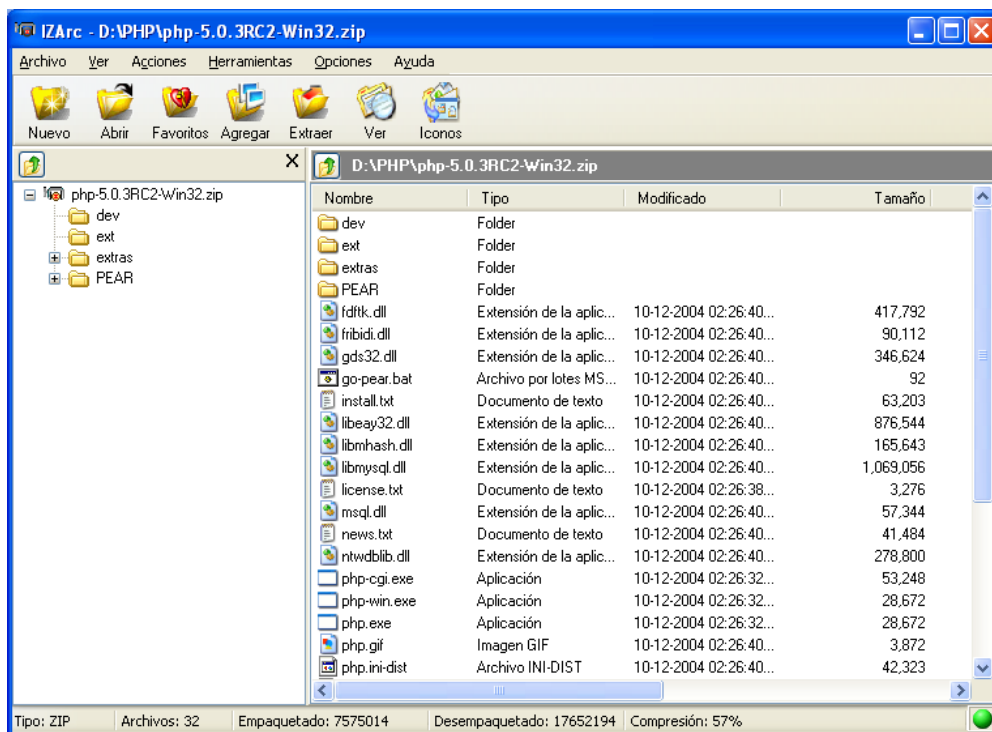
en esta página debemos bajar dos archivos, el primero es PHP 5.0.3 installer y el segundo archivo es PHP 5.0.3 zip package



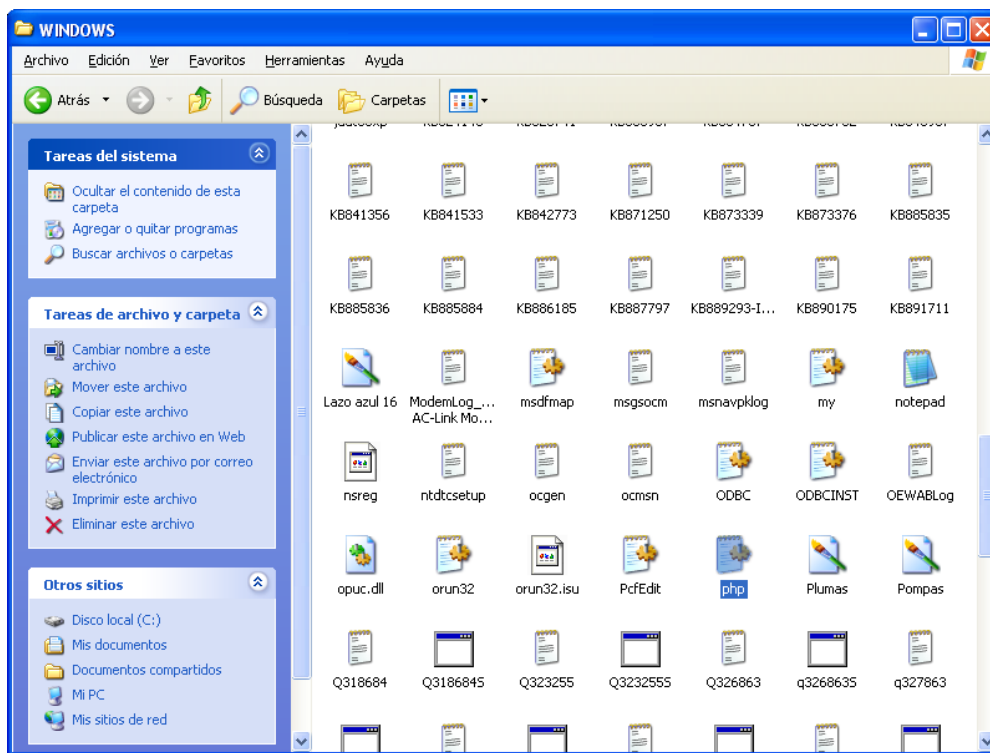
ya que se tienen ambos archivos en nuestro equipo ejecutamos en instalador con las opciones por default, lo único que debemos modificar es cuando nos pide que indiquemos el tipo de servidor que usamos seleccionamos Apache:



una vez terminada la instalación se descomprime el archivo PHP 5.0.3 zip package y sobrescribimos todos los archivos en donde se realizó la instalación (por default debe quedar en C:\PHP) sobrescribiendo absolutamente todos los archivos

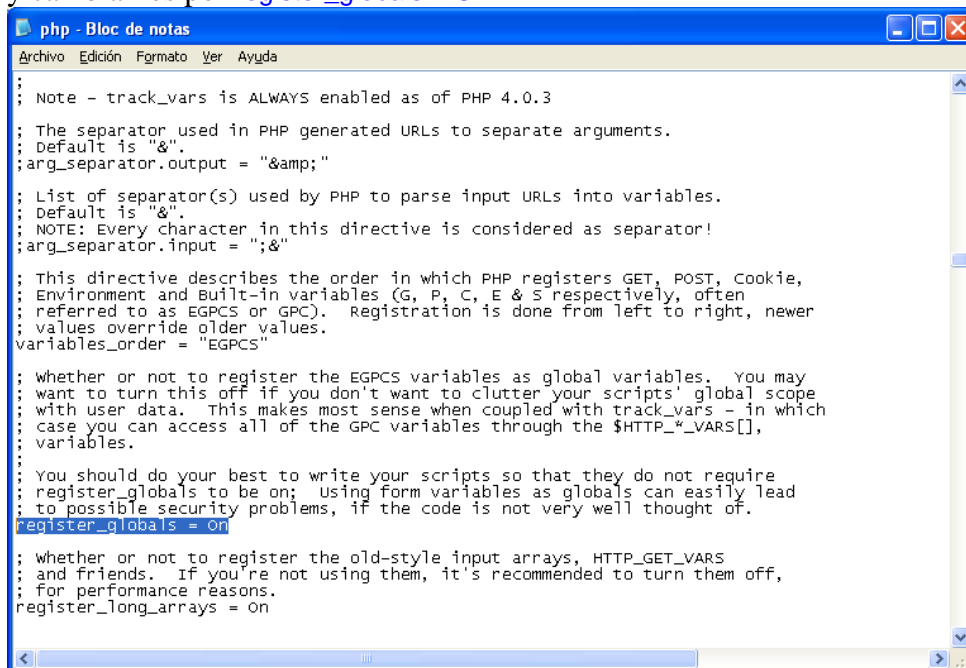


ahora lo que falta es modificar el archivo PHP.SYS que se encuentra en C:\windows\

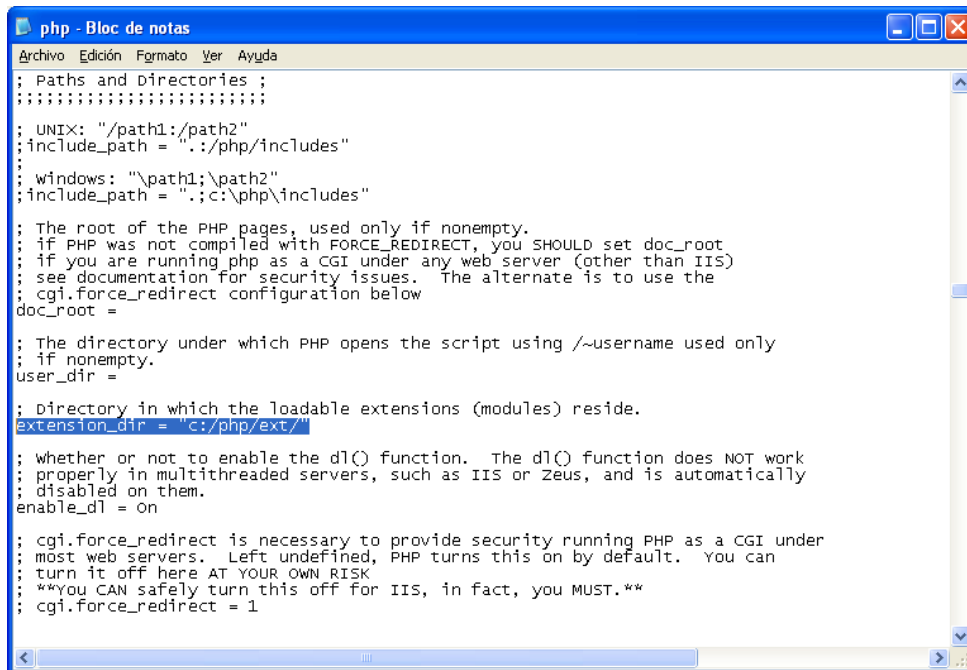


donde buscamos la cadena `register_globals = Off`

y cambiamos por `register_globals = On`



posteriormente buscamos la cadena `extension_dir`, donde escribiremos `extension_dir = "c:/php/ext/"`



```
php - Bloc de notas
Archivo Edición Formato Ver Ayuda

; Paths and Directories ;
;::::::::::::::::::::::::;

; UNIX: "/path1:/path2"
;include_path = "./php/includes"
;
; windows: "\\path1;\path2"
;include_path = ".;c:\php\includes"
;

; The root of the PHP pages, used only if nonempty.
; if PHP was not compiled with FORCE_REDIRECT, you SHOULD set doc_root
; if you are running php as a CGI under any web server (other than IIS)
; see documentation for security issues.  The alternate is to use the
; cgi.force_redirect configuration below
doc_root =

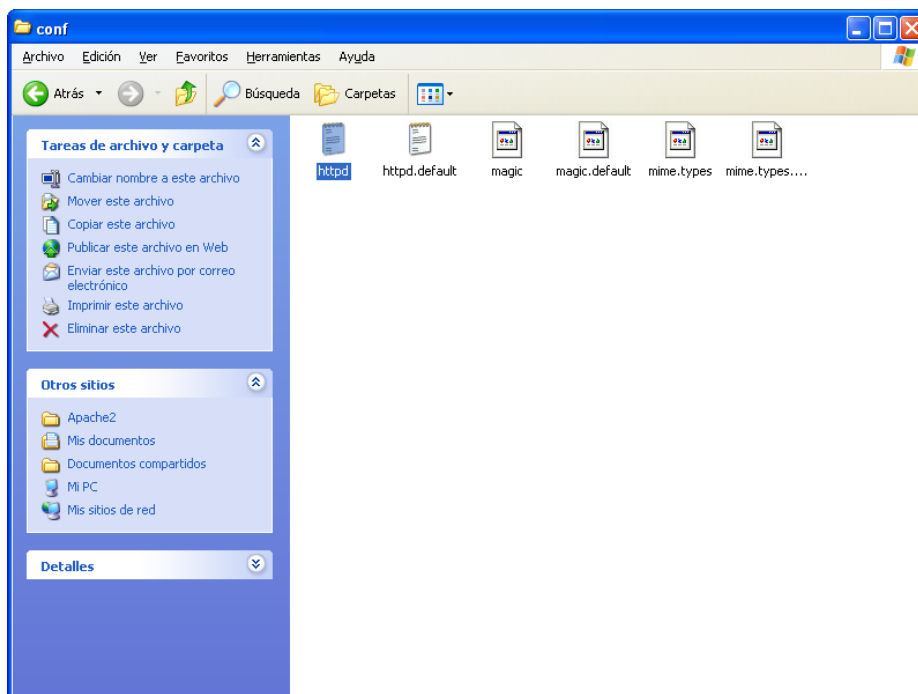
; The directory under which PHP opens the script using /~username used only
; if nonempty.
user_dir =

; Directory in which the loadable extensions (modules) reside.
extension_dir = "c:/php/ext/"

; whether or not to enable the dl() function.  The dl() function does NOT work
; properly in multithreaded servers, such as IIS or Zeus, and is automatically
; disabled on them.
enable_dl = On

; cgi.force_redirect is necessary to provide security running PHP as a CGI under
; most web servers.  Left undefined, PHP turns this on by default.  You can
; turn it off here AT YOUR OWN RISK
; **you CAN safely turn this off for IIS, in fact, you MUST.**
; cgi.force_redirect = 1
```

ya que hicimos los cambios guardamos el archivo y lo cerramos, ahora lo que debemos hacer es buscar el archivo C:\Archivos de programa\Apache Group\Apache2\conf\httpd

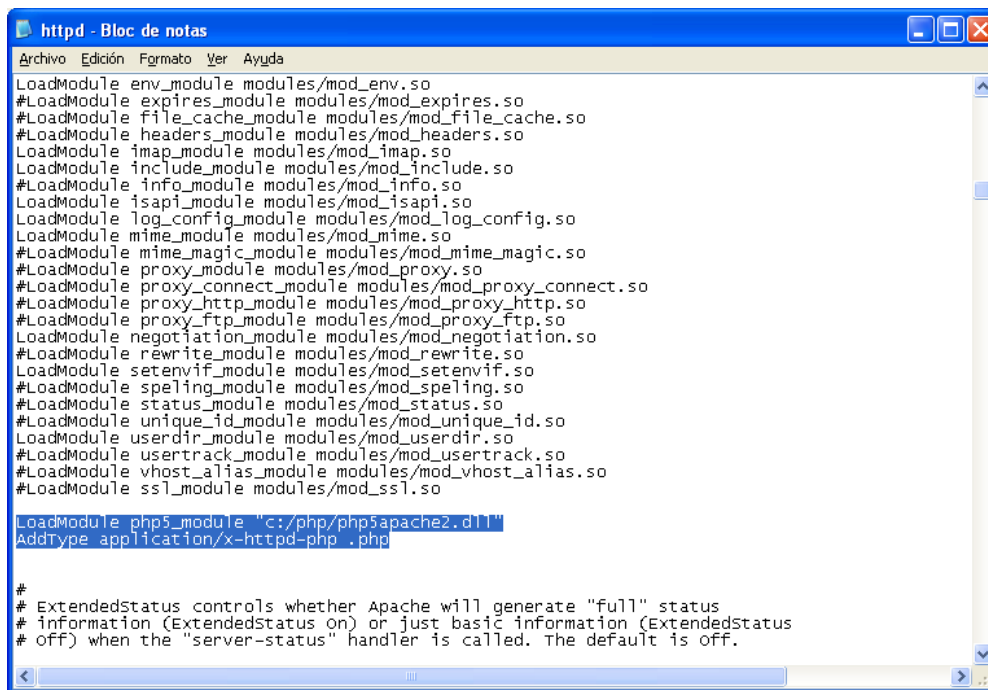


Buscamos "Dynamic Shared Object (DSO) Support", que es donde se cargan los módulos. Y encontramos hasta un ejemplo:

```
# Example:
# LoadModule foo_module modules/mod_foo.so
#
```

y añadimos:

```
LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php
```



```
httpd - Bloc de notas
Archivo  Edición  Formato  Ver  Ayuda
LoadModule env_module modules/mod_env.so
#LoadModule expires_module modules/mod_expires.so
#LoadModule file_cache_module modules/mod_file_cache.so
#LoadModule headers_module modules/mod_headers.so
LoadModule imap_module modules/mod_imap.so
LoadModule include_module modules/mod_include.so
#LoadModule info_module modules/mod_info.so
LoadModule isapi_module modules/mod_isapi.so
LoadModule log_config_module modules/mod_log_config.so
LoadModule mime_module modules/mod_mime.so
#LoadModule mime_magic_module modules/mod_mime_magic.so
#LoadModule proxy_module modules/mod_proxy.so
#LoadModule proxy_connect_module modules/mod_proxy_connect.so
#LoadModule proxy_http_module modules/mod_proxy_http.so
#LoadModule proxy_ftp_module modules/mod_proxy_ftp.so
LoadModule negotiation_module modules/mod_negotiation.so
#LoadModule rewrite_module modules/mod_rewrite.so
LoadModule setenvif_module modules/mod_setenvif.so
#LoadModule spelling_module modules/mod_spelling.so
#LoadModule status_module modules/mod_status.so
#LoadModule unique_id_module modules/mod_unique_id.so
LoadModule userdir_module modules/mod_userdir.so
#LoadModule usertrack_module modules/mod_usertrack.so
#LoadModule vhost_alias_module modules/mod_vhost_alias.so
#LoadModule ssl_module modules/mod_ssl.so

LoadModule php5_module "c:/php/php5apache2.dll"
AddType application/x-httpd-php .php

#
# ExtendedStatus controls whether Apache will generate "full" status
# information (ExtendedStatus on) or just basic information (ExtendedStatus
# off) when the "server-status" handler is called. The default is off.
```

Ahora buscamos el DirectoryIndex:

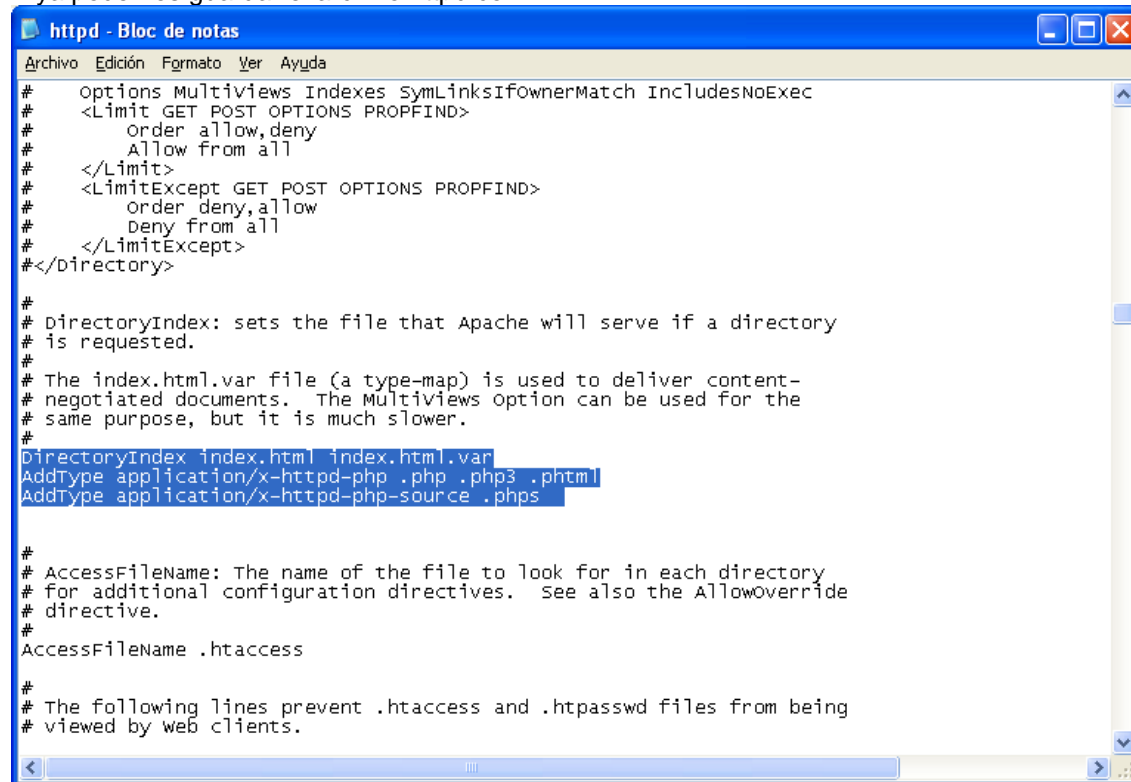
[DirectoryIndex index.html index.html.var](#)

Ahora le añadimos estas líneas:

[AddType application/x-httpd-php .php .php3 .phtml](#)

[AddType application/x-httpd-php-source .phps](#)

Y ya podemos guardar el archivo httpd.conf



```
# Options Multiviews Indexes SymLinksIfOwnerMatch IncludesNoExec
# <Limit GET POST OPTIONS PROPFIND>
#     Order allow,deny
#     Allow from all
# </Limit>
# <LimitExcept GET POST OPTIONS PROPFIND>
#     Order deny,allow
#     Deny from all
# </LimitExcept>
#</Directory>

#
# DirectoryIndex: sets the file that Apache will serve if a directory
# is requested.
#
# The index.html.var file (a type-map) is used to deliver content-
# negotiated documents. The Multiviews Option can be used for the
# same purpose, but it is much slower.
#
DirectoryIndex index.html index.html.var
AddType application/x-httpd-php .php .php3 .phtml
AddType application/x-httpd-php-source .phps

#
# AccessFileName: The name of the file to look for in each directory
# for additional configuration directives. See also the AllowOverride
# directive.
#
AccessFileName .htaccess

#
# The following lines prevent .htaccess and .htpasswd files from being
# viewed by web clients.
```

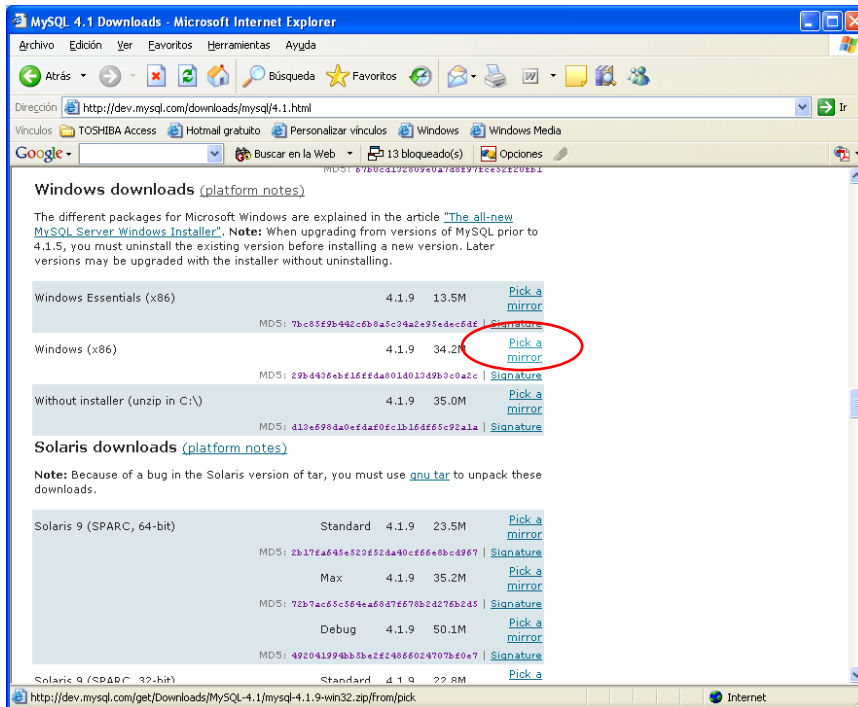
Una vez hechos los cambios solo resta guardar el archivo y de preferencia reiniciar los servicios de Apache o en su defecto reiniciar nuestro equipo para que hagan efecto los cambios.

En php.ini hay que quitarle el comentario a la línea 'extension=php_mysql.dll' y copiar el archivo php_mysql.dll del directorio c:\php\ext\ al directorio de Windows.

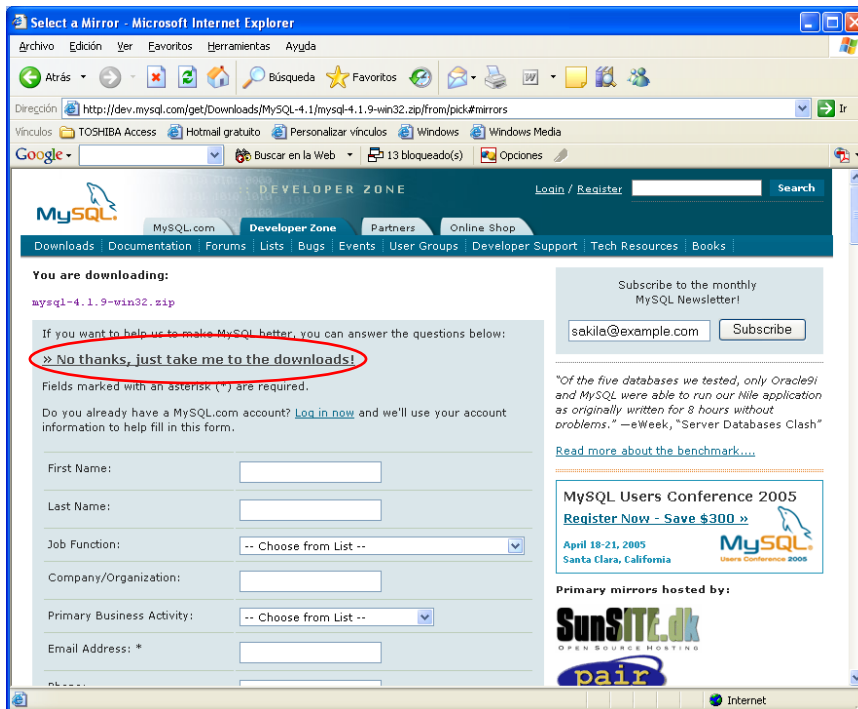
2.4 Instalación y configuración de MySQL

A continuación se describe el procedimiento para descargar e instalar el servidor MySQL.

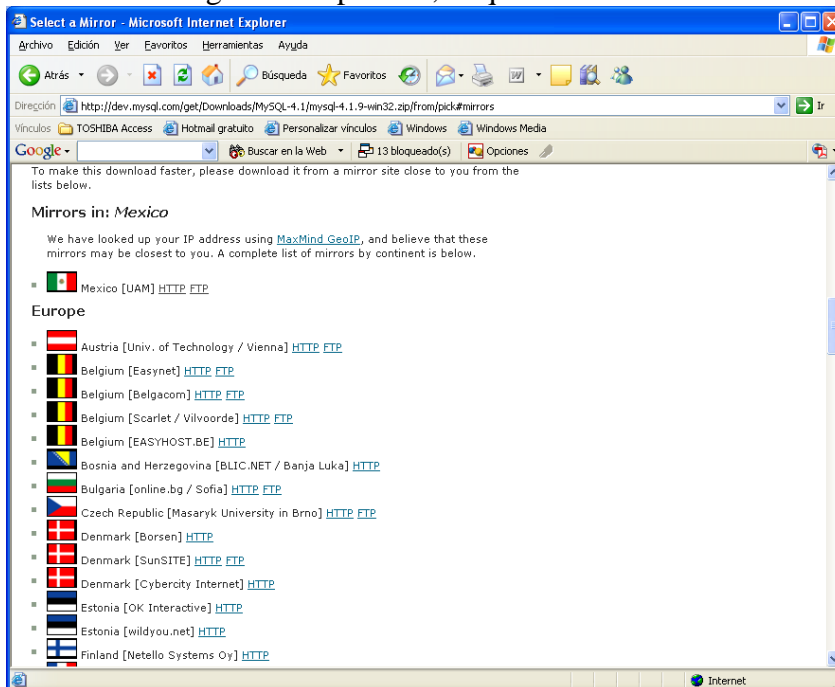
1. Primero hay que bajar el archivo de instalación, en:
<http://dev.mysql.com/downloads/mysql/4.1.html>
2. Se desplegará la página de downloads de Mysql. Es necesario bajar hasta encontrar la sección “Windows downloads”. Luego, hacer clic sobre la etiqueta “Pick a mirror” correspondiente a Windows(x86):



3. Lo anterior conducirá a la siguiente pantalla:



4. Ahí el registro es opcional, lo que realmente interesa es bajar el programa:



5. Se puede seleccionar cualquier liga de las anteriores, siempre y cuando funcione. En lo particular, la bajé en:

-  United States [Cloud 9 Internet / White Plains, NY] [HTTP](#)

Se abrirá la ventana estándar de windows preguntando donde guardar el archivo mysql-4.1.9-win32.zip.

6. Una vez bajado el archivo anterior, es necesario descomprimirlo en alguna carpeta del disco duro. Y enseguida, se debe ejecutar el archivo setup.exe.
7. A partir de ahí se puede realizar la instalación por default sin mayor problema. Al llegar a la pantalla en la que se debe seleccionar el tipo de instalación, elegir la "Típica".

En caso de que el archivo bajado no contenga un archivo setup.exe, únicamente es necesario descomprimir su contenido en el directorio c:\mysql\

2.5 Administración de bases de datos MySQL

En esta sección se describe el uso del programa cliente mysql para crear y usar una sencilla base de datos. Los códigos y la descripción que los acompañan fueron tomados de [5].

Mysql (algunas veces referido como "monitor mysql") es un programa interactivo que permite conectarnos a un servidor MySQL, ejecutar algunas consultas, y ver los resultados. mysql puede ser usado también en modo batch: es decir, se pueden colocar toda una serie de consultas en un archivo, y posteriormente decirle a mysql que ejecute dichas consultas. Para ver la lista de opciones proporcionadas por mysql, lo invocamos con la opción --help:

```
shell> mysql --help
```

A continuación se describe el proceso completo de creación y uso de una base de datos en MySQL [4][6][7]. Si se está interesado sólo en el acceso y uso de una base de datos existente, se pueden omitir las secciones que describen como crear la base de datos y las tablas correspondientes.

Puesto que es imposible que se describan a detalle muchos de los tópicos cubiertos aquí, se recomienda que se consulte el manual de MySQL para obtener más información al respecto.

Conectándose y desconectándose al servidor MySQL

Para conectarse al servidor, usualmente necesitamos de un nombre de usuario (login) y de una contraseña (password), y si el servidor al que nos deseamos conectar está en una máquina diferente de la nuestra, también necesitamos indicar el nombre o la dirección IP de dicho servidor. Una vez que conocemos estos tres valores, podemos conectarnos de la siguiente manera:

```
shell> mysql -h NombreDelServidor -u NombreDeUsuario -p
```

Cuando ejecutamos este comando, se nos pedirá que proporcionemos también la contraseña para el nombre de usuario que estamos usando.

Si la conexión al servidor MySQL se pudo establecer de manera satisfactoria, recibiremos el mensaje de bienvenida y estaremos en el prompt de mysql:

```

shell>mysql -h casita -u root -p

Enter password: *****

Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5563 to server version: 3.23.41

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>

```

Este prompt nos indica que mysql está listo para recibir comandos.

Algunas instalaciones permiten que los usuarios se conecten de manera anónima al servidor corriendo en la máquina local. Si es el caso de nuestra máquina, debemos de ser capaces de conectarnos al servidor invocando a mysql sin ninguna opción:

```

shell> mysql

```

Después de que nos hemos conectado de manera satisfactoria, podemos desconectarnos en cualquier momento al escribir "quit", "exit", o presionar CONTROL+D.

La mayoría de los ejemplos siguientes asume que estamos conectados al servidor, lo cual se indica con el prompt de mysql.

Ejecutando algunas consultas

En este momento debimos de haber podido conectarnos ya al servidor MySQL, aún cuando no hemos seleccionado alguna base de datos para trabajar. Lo que haremos a continuación es escribir algunos comandos para irnos familiarizando con el funcionamiento de mysql

```

mysql> SELECT VERSION(), CURRENT_DATE;
+-----+-----+
| VERSION() | CURRENT_DATE |
+-----+-----+
| 3.23.41   | 2002-10-01   |
+-----+-----+
1 row in set (0.03 sec)

mysql>

```

Esta comando ilustra distintas cosas acerca de mysql:

- Un comando normalmente consiste de un sentencia SQL seguida por un punto y coma.
- Cuando emitimos un comando, mysql lo manda al servidor para que lo ejecute, nos muestra los resultados y regresa el prompt indicando que está listo para recibir más consultas.
- mysql muestra los resultados de la consulta como una tabla (filas y columnas). La primera fila contiene etiquetas para las columnas. Las filas siguientes muestran los resultados de la consulta. Normalmente las etiquetas de las columnas son los nombres de los campos de las tablas que estamos usando en alguna consulta. Si lo que estamos recuperando es el valor de una expresión (como en el ejemplo anterior) las etiquetas en las columnas son la expresión en sí.
- mysql muestra cuántas filas fueron regresadas y cuanto tiempo tardó en ejecutarse la consulta, lo cual puede darnos una idea de la eficiencia del servidor, aunque estos valores pueden ser un tanto imprecisos ya que no se muestra la hora del CPU, y

porque pueden verse afectados por otros factores, tales como la carga del servidor y la velocidad de comunicación en una red.

- Las palabras clave pueden ser escritas usando mayúsculas y minúsculas.

Las siguientes consultas son equivalentes:

```
mysql> SELECT VERSION(), CURRENT_DATE;
mysql> select version(), current_date;
mysql> SeLeCt vErSiOn(), current_DATE;
```

Aquí está otra consulta que demuestra como se pueden escribir algunas expresiones matemáticas y trigonométricas:

```
mysql> SELECT SIN(PI()/4), (4+1)*5;
+-----+-----+
| SIN(PI()/4) | (4+1)*5 |
+-----+-----+
| 0.707107 | 25 |
+-----+-----+
```

Aunque hasta este momento se han escrito sentencias sencillas de una sólo línea, es posible escribir más de una sentencia por línea, siempre y cuando estén separadas por punto y coma:

```
mysql> SELECT VERSION(); SELECT NOW();
+-----+
| VERSION() |
+-----+
| 3.23.41 |
+-----+
1 row in set (0.01 sec)

+-----+
| NOW() |
+-----+
| 2002-10-28 14:26:04 |
+-----+
1 row in set (0.01 sec)
```

Un comando no necesita ser escrito en una sólo línea, así que los comandos que requieran de varias líneas no son un problema. mysql determinará en donde finaliza la sentencia cuando encuentre el punto y coma, no cuando encuentre el fin de línea.

Aquí está un ejemplo que muestra un consulta simple escrita en varias líneas:

```
mysql> SELECT
-> USER(),
-> CURRENT_DATE;
+-----+-----+
| USER() | CURRENT_DATE |
+-----+-----+
| root@localhost | 2002-09-14 |
+-----+-----+
1 row in set (0.00 sec)

mysql>
```

En este ejemplo debe notarse como cambia el prompt (de mysql> a ->) cuando se escribe una consulta en varias líneas. Esta es la manera en cómo mysql indica que está esperando a que finalice la consulta. Sin embargo si deseamos no terminar de escribir la consulta, podemos hacerlo al escribir \c como se muestra en el siguiente ejemplo:

```
mysql> SELECT
-> USER(),
-> \c
mysql>
```

De nuevo, se nos regresa el comando el prompt `mysql>` que nos indica que `mysql` está listo para una nueva consulta.

En la siguiente tabla se muestran cada uno de los prompts que podemos obtener y una breve descripción de su significado para `mysql` [5]:

Prompt	Significado
<code>mysql></code>	Listo para una nueva consulta.
<code>-></code>	Esperando la línea siguiente de una consulta multi-línea.
<code>'></code>	Esperando la siguiente línea para completar una cadena que comienza con una comilla sencilla (').
<code>"></code>	Esperando la siguiente línea para completar una cadena que comienza con una comilla doble (").

Los comandos multi-línea comúnmente ocurren por accidente cuando tecleamos ENTER, pero olvidamos escribir el punto y coma. En este caso `mysql` se queda esperando para que finalicemos la consulta:

```
mysql> SELECT USER()  
->
```

Si esto llega a suceder, muy probablemente `mysql` estará esperando por un punto y coma, de manera que si escribimos el punto y coma podremos completar la consulta y `mysql` podrá ejecutarla:

```
mysql> SELECT USER()  
-> ;  
+-----+  
| USER() |  
+-----+  
| root@localhost |  
+-----+  
1 row in set (0.00 sec)  
  
mysql>
```

Los prompts `'>` y `">` ocurren durante la escritura de cadenas. En `mysql` podemos escribir cadenas utilizando comillas sencillas o comillas dobles (por ejemplo, `'hola'` y `"hola"`), y `mysql` nos permite escribir cadenas que ocupen multiple líneas. De manera que cuando veamos el prompt `'>` o `">`, `mysql` nos indica que hemos empezado a escribir una cadena, pero no la hemos finalizado con la comilla correspondiente.

Aunque esto puede suceder si estamos escribiendo una cadena muy grande, es más frecuente que obtengamos alguno de estos prompts si inadvertidamente escribimos alguna de estas comillas.

Por ejemplo:

```
mysql> SELECT * FROM mi_tabla WHERE nombre = "Lupita AND edad < 30;  
">
```

Si escribimos esta consulta `SELECT` y entonces presionamos ENTER para ver el resultado, no sucederá nada. En lugar de preocuparnos porque la consulta ha tomado mucho tiempo, debemos notar la pista que nos da `mysql` cambiando el prompt. Esto nos indica que `mysql` está esperando que finalicemos la cadena iniciada (`"Lupita`).

En este caso, ¿qué es lo que debemos hacer? . La cosa más simple es cancelar la consulta. Sin embargo, no basta con escribir `\c`, ya que `mysql` interpreta esto como parte de la cadena

que estamos escribiendo. En lugar de esto, debemos escribir antes la comilla correspondiente y después \c :

```
mysql> SELECT * FROM mi_tabla WHERE nombre = "Lupita AND edad < 30;
"> " \c
mysql>
```

El prompt cambiará de nuevo al ya conocido mysql>, indicándonos que mysql está listo para una nueva consulta.

Es sumamente importante conocer lo que significan los prompts '>' y '>', ya que si en algún momento nos aparece alguno de ellos, todas las líneas que escribamos a continuación serán consideradas como parte de la cadena, inclusive cuando escribamos QUIT. Esto puede ser confuso, especialmente si no sabemos que es necesario escribir la comilla correspondiente para finalizar la cadena, para que podamos escribir después algún otro comando, o terminar la consulta que deseamos ejecutar.

Creando y usando una base de datos

Ahora que conocemos como escribir y ejecutar sentencias, es tiempo de acceder a una base de datos.

Supongamos que tenemos diversas mascotas en casa (nuestro pequeño zoológico) y deseamos tener registros de los datos acerca de ellas. Podemos hacer esto al crear tablas que guarden esta información, para que posteriormente la consulta de estos datos sea bastante fácil y de manera muy práctica. Esta sección muestra como crear una base de datos, crear una tabla, incorporar datos en una tabla, y recuperar datos de las tablas de diversas maneras.

La base de datos "zoológico" será muy simple (deliberadamente), pero no es difícil pensar de situaciones del mundo real en la cual una base de datos similar puede ser usada.

Primeramente usaremos la sentencia SHOW para ver cuáles son las bases de datos existentes en el servidor al que estamos conectados:

```
mysql> SHOW DATABASES;
+-----+
| Database |
+-----+
| mysql    |
| test     |
+-----+
2 rows in set (0.00 sec)

mysql>
```

Es probable que la lista de bases de datos que veamos sea diferente en nuestro caso, pero seguramente las bases de datos "mysql" y "test" estarán entre ellas. En particular, la base de datos "mysql" es requerida, ya que ésta tiene la información de los privilegios de los usuarios de MySQL. La base de datos "test" es creada durante la instalación de MySQL con el propósito de servir como área de trabajo para los usuarios que inician en el aprendizaje de MySQL.

Se debe anotar también que es posible que no veamos todas las bases de datos si no tenemos el privilegio SHOW DATABASES. Se recomienda revisar la sección del manual de MySQL dedicada a los comandos GRANT y REVOKE.

Si la base de datos "test" existe, hay que intentar acceder a ella:

```
mysql> USE test
Database changed
```

```
mysql>
```

Observar que USE, al igual que QUIT, no requieren el uso del punto y coma, aunque si se usa éste, no hay ningún problema. El comando USE es especial también de otra manera: éste debe ser usado en una sólo línea.

Podríamos usar la base de datos "test" (si tenemos acceso a ella) para los ejemplos que vienen a continuación, pero cualquier cosa que hagamos puede ser eliminada por cualquier otro usuario que tenga acceso a esta base de datos. Por esta razón, es recomendable que preguntemos al administrador MySQL acerca de la base de datos que podemos usar. Supongamos que deseamos tener una base de datos llamada "zoologico" (nótese que no se está acentuando la palabra) a la cual sólo nosotros tengamos acceso, para ello el administrador necesita ejecutar un comando como el siguiente:

```
mysql> GRANT ALL on zoologico.* TO MiNombreUsuario@MiComputadora
-> IDENTIFIED BY 'MiContraseña';
```

en donde MiNombreUsuario es el nombre de usuario asignado dentro del contexto de MySQL, MiComputadora es el nombre o la dirección IP de la computadora desde la que nos conectamos al servidor MySQL, y MiContraseña es la contraseña que se nos ha asignado, igualmente, dentro del ambiente de MySQL exclusivamente. Ambos, nombre de usuario y contraseña no tienen nada que ver con el nombre de usuario y contraseña manejados por el sistema operativo (si es el caso).

Si el administrador creó la base de datos al momento de asignar los permisos, podemos hacer uso de ella. De otro modo, nosotros debemos crearla:

```
mysql> USE zoologico
ERROR 1049: Unknown database 'zoologico'
mysql>
```

El mensaje anterior indica que la base de datos no ha sido creada, por lo tanto necesitamos crearla.

```
mysql> CREATE DATABASE zoologico;
Query OK, 1 row affected (0.00 sec)
```

```
mysql> USE zoologico
Database changed
mysql>
```

Bajo el sistema operativo Unix, los nombres de las bases de datos son sensibles al uso de mayúsculas y minúsculas (no como las palabras clave de SQL), por lo tanto debemos de tener cuidado de escribir correctamente el nombre de la base de datos. Esto es cierto también para los nombres de las tablas.

Al crear una base de datos no se selecciona ésta de manera automática; debemos hacerlo de manera explícita, por ello usamos el comando USE en el ejemplo anterior.

La base de datos se crea sólo una vez, pero nosotros debemos seleccionarla cada vez que iniciamos una sesión con mysql. Por ello es recomendable que se indique la base de datos sobre la que vamos a trabajar al momento de invocar al monitor de MySQL. Por ejemplo:

```
shell>mysql -h casita -u bluman -p zoologico

Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 17 to server version: 3.23.38-nt

Type 'help;' or '\h' for help. Type '\c' to clear the buffer

mysql>
```

Observar que "zoologico" no es la contraseña que se está proporcionando desde la línea de comandos, sino el nombre de la base de datos a la que deseamos acceder. Si deseamos proporcionar la contraseña en la línea de comandos después de la opción "-p", debemos de hacerlo sin dejar espacios (por ejemplo, -phola123, no como -p hola123). Sin embargo, escribir nuestra contraseña desde la línea de comandos no es recomendado, ya que es bastante inseguro.

Creando una tabla

Crear la base de datos es la parte más fácil, pero en este momento la base de datos está vacía, como lo indica el comando SHOW TABLES:

```
mysql> SHOW TABLES;
Empty set (0.00 sec)
```

La parte un tanto complicada es decidir la estructura que debe tener nuestra base de datos: qué tablas se necesitan y qué columnas estarán en cada tabla.

En principio, necesitamos una tabla que contenga un registro para cada una de nuestras mascotas. Ésta puede ser una tabla llamada mascotas, y debe contener por lo menos el nombre de cada uno de nuestros animalitos. Ya que el nombre en sí no es muy interesante, la tabla debe contener alguna otra información. Por ejemplo, si más de una persona en nuestra familia tiene una mascota, es probable que tengamos que guardar la información acerca de quien es el dueño de cada mascota. Así mismo, también sería interesante contar con alguna información más descriptiva tal como la especie, y el sexo de cada mascota.

¿Y que sucede con la edad?. Esto puede ser también de interés, pero no es una buena idea almacenar este dato en la base de datos. La edad cambia conforme pasa el tiempo, lo cual significa que debemos de actualizar los registros frecuentemente. En vez de esto, es una mejor idea guardar un valor fijo, tal como la fecha de nacimiento. Entonces, cuando necesitemos la edad, la podemos calcular como la diferencia entre la fecha actual y la fecha de nacimiento. MySQL proporciona funciones para hacer operaciones entre fechas, así que no hay ningún problema.

Al almacenar la fecha de nacimiento en lugar de la edad tenemos algunas otras ventajas:

Podemos usar la base de datos para tareas tales como generar recordatorios para cada cumpleaños próximo de nuestras mascotas. Podemos calcular la edad en relación a otras fechas que la fecha actual. Por ejemplo, si almacenamos la fecha en que murió nuestra mascota en la base de datos, es fácil calcular que edad tenía nuestro animalito cuando falleció. Es probable que estemos pensando en otro tipo de información que sería igualmente útil en la tabla "mascotas", pero para nosotros será suficiente por ahora contar con información de nombre, propietario, especie, nacimiento y fallecimiento.

Usaremos la sentencia CREATE TABLE para indicar como estarán conformados los registros de nuestras mascotas.

```
mysql> CREATE TABLE mascotas(
-> nombre VARCHAR(20), propietario VARCHAR(20),
-> especie VARCHAR(20), sexo CHAR(1), nacimiento DATE,
-> fallecimiento DATE);
Query OK, 0 rows affected (0.02 sec)

mysql>
```

VARCHAR es una buena elección para los campos nombre, propietario, y especie, ya que los valores que almacenarán son de longitud variable. No es necesario que la longitud de estas columnas sea la misma, ni tampoco que sea de 20. Se puede especificar cualquier longitud entre 1 y 255, lo que se considere más adecuado. Si resulta que la elección de la longitud de los campos que hemos hecho no resultó adecuada, MySQL proporciona una sentencia ALTER TABLE que nos puede ayudar a solventar este problema.

El campo sexo puede ser representado en una variedad de formas, por ejemplo, "m" y "f", o tal vez "masculino" y "femenino", aunque resulta más simple la primera opción.

El uso del tipo de dato DATE para los campos nacimiento y fallecimiento debe de resultar obvio.

Ahora que hemos creado la tabla, la sentencia SHOW TABLES debe producir algo como:

```
mysql> SHOW TABLES;
+-----+
| Tables_in_zoologico |
+-----+
| mascotas             |
+-----+
1 row in set (0.00 sec)

mysql>
```

Para verificar que la tabla fué creada como nosotros esperabamos, usaremos la sentencia DESCRIBE:

```
mysql> DESCRIBE mascotas;
+-----+-----+-----+-----+-----+-----+
| Field      | Type      | Null | Key | Default | Extra |
+-----+-----+-----+-----+-----+-----+
| nombre     | varchar(20) | YES  |     | NULL    |       |
| propietario | varchar(20) | YES  |     | NULL    |       |
| especie    | varchar(20) | YES  |     | NULL    |       |
| sexo       | char(1)    | YES  |     | NULL    |       |
| nacimiento | date       | YES  |     | NULL    |       |
| fallecimiento | date       | YES  |     | NULL    |       |
+-----+-----+-----+-----+-----+-----+
6 rows in set (0.01 sec)

mysql>
```

Podemos hacer uso de la sentencia DESCRIBE en cualquier momento, por ejemplo, si olvidamos los nombres ó el tipo de las columnas en la tabla.

Cargando datos en una tabla

Después de haber creado la tabla, ahora podemos incorporar algunos datos en ella, para lo cual haremos uso de las sentencias INSERT y LOAD DATA.

Supongamos que los registros de nuestras mascotas pueden ser descritos por los datos mostrados en la siguiente tabla.

Nombre	Propietario	Especie	Sexo	Nacimiento	Fallecimiento
Fluffy	Arnoldo	Gato	f	1999-02-04	
Mau	Juan	Gato	m	1998-03-17	
Buffy	Arnoldo	Perro	f	1999-05-13	
FanFan	Benito	Perro	m	2000-08-27	
Kaiser	Diana	Perro	m	1998-08-31	1997-07-29

Chispa	Omar	Ave	f	1998-09-11	
Wicho	Tomás	Ave		2000-02-09	
Skim	Benito	Serpiente	m	2001-04-29	

Debemos observar que MySQL espera recibir fechas en el formato YYYY-MM-DD, que puede ser diferente a lo que nosotros estamos acostumbrados.

Ya que estamos iniciando con una tabla vacía, la manera más fácil de poblarla es crear un archivo de texto que contenga un registro por línea para cada uno de nuestros animalitos para que posteriormente carguemos el contenido del archivo en la tabla únicamente con una sentencia.

Por tanto, debemos de crear un archivo de texto "mascotas.txt" que contenga un registro por línea con valores separados por tabuladores, cuidando que el orden de las columnas sea el mismo que utilizamos en la sentencia CREATE TABLE. Para valores que no conozcamos podemos usar valores nulos (NULL). Para representar estos valores en nuestro archivo debemos usar \N.

Para cargar el contenido del archivo en la tabla mascotas, usaremos el siguiente comando:

```
mysql> LOAD DATA LOCAL INFILE "mascotas.txt" INTO TABLE mascotas;
```

La sentencia LOAD DATA nos permite especificar cuál es el separador de columnas, y el separador de registros, por default el tabulador es el separador de columnas (campos), y el salto de línea es el separador de registros, que en este caso son suficientes para que la sentencia LOAD DATA lea correctamente el archivo "mascotas.txt".

Si lo que deseamos es añadir un registro a la vez, entonces debemos hacer uso de la sentencia INSERT. En la manera más simple, debemos proporcionar un valor para cada columna en el orden en el cual fueron listados en la sentencia CREATE TABLE. Supongamos que nuestra hermana Diana compra un nuevo hamster nombrado Pelusa. Podemos usar la sentencia INSERT para agregar su registro en nuestra base de datos.

```
mysql> INSERT INTO mascotas
-> VALUES('Pelusa','Diana','Hamster','f','2000-03-30',NULL);
```

Notar que los valores de cadenas y fechas deben estar encerrados entre comillas. También, con la sentencia INSERT podemos insertar el valor NULL directamente para representar un valor nulo, un valor que no conocemos. En este caso no se usa \N como en el caso de la sentencia LOAD DATA.

De este ejemplo, debemos ser capaces de ver que es un poco más la tarea que se tiene que realizar si inicialmente cargamos los registros con varias sentencias INSERT en lugar de una única sentencia LOAD DATA.

Recuperando información de una tabla

La sentencia SELECT es usada para obtener la información guardada en una tabla. La forma general de esta sentencia es:

```
SELECT LaInformaciónQueDeseamos FROM DeQueTabla WHERE CondiciónASatisfacer
```

Aquí, LaInformaciónQueDeseamos es la información que queremos ver. Esta puede ser una lista de columnas, o un * para indicar "todas las columnas". DeQueTabla indica el nombre de la tabla de la cual vamos a obtener los datos. La cláusula WHERE es opcional. Si está

presente, la Condición ASatisfacer especifica las condiciones que los registros deben satisfacer para que puedan ser mostrados.

Seleccionando todos los datos

La manera más simple de la sentencia SELECT es cuando se recuperan todos los datos de una tabla:

```
mysql> SELECT * FROM mascotas;
```

nombre	propietario	especie	sexo	nacimiento	fallecimiento
Fluffy	Arnoldo	Gato	f	1999-02-04	NULL
Mau	Juan	Gato	m	1998-03-17	NULL
Buffy	Arnoldo	Perro	f	1999-05-13	NULL
FanFan	Benito	Perro	m	2000-08-27	NULL
Kaiser	Diana	Perro	m	1998-08-31	1997-07-29
Chispa	Omar	Ave	f	1998-09-11	NULL
Wicho	Tomás	Ave	NULL	2000-02-09	NULL
Skim	Benito	Serpiente	m	2001-04-29	NULL
Pelusa	Diana	Hamster	f	2000-03-30	NULL

```
9 rows in set (0.00 sec)
```

Esta forma del SELECT es útil si deseamos ver los datos completos de la tabla, por ejemplo, para asegurarnos de que están todos los registros después de la carga de un archivo.

Por ejemplo, en este caso que estamos tratando, al consultar los registros de la tabla, nos damos cuenta de que hay un error en el archivo de datos (mascotas.txt): parece que Kaiser ha nacido después de que ha fallecido!. Al revisar un poco el pedigree de Kaiser encontramos que la fecha correcta de nacimiento es el año 1989, no 1998.

Hay por lo menos un par de maneras de solucionar este problema:

Editar el archivo "mascotas.txt" para corregir el error, eliminar los datos de la tabla mascotas con la sentencia DELETE, y cargar los datos nuevamente con el comando LOAD DATA:

```
mysql> DELETE FROM mascotas;
mysql> LOAD DATA LOCAL INFILE "mascotas.txt" INTO TABLE mascotas;
```

Sin embargo, si hacemos esto, debemos ingresar los datos de Pelusa, la mascota de nuestra hermana Diana.

La segunda opción consiste en corregir sólo el registro erróneo con una sentencia UPDATE:

```
mysql> UPDATE mascotas SET nacimiento="1989-08-31" WHERE nombre="Kaiser";
```

Como se mostró anteriormente, es muy fácil recuperar los datos de una tabla completa. Pero típicamente no deseamos hacer esto, particularmente cuando las tablas son demasiado grandes. En vez de ello, estaremos más interesados en responder preguntas particulares, en cuyo caso debemos especificar algunas restricciones para la información que deseamos ver.

Seleccionando registros particulares

Podemos seleccionar sólo registros particulares de una tabla. Por ejemplo, si deseamos verificar el cambio que hicimos a la fecha de nacimiento de Kaiser, seleccionamos sólo el registro de Kaiser de la siguiente manera:

```
mysql> SELECT * FROM mascotas WHERE nombre="Kaiser";
```

```
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Kaiser | Diana      | Perro   | m    | 1989-08-31 | 1997-07-29    |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

La salida mostrada confirma que el año ha sido corregido de 1998 a 1989.

La comparación de cadenas es normalmente no sensitiva, así que podemos especificar el nombre como "kaiser", "KAISER", etc. El resultado de la consulta será el mismo.

Podemos además especificar condiciones sobre cualquier columna, no sólo el "nombre". Por ejemplo, si deseamos conocer qué mascotas nacieron después del 2000, tendríamos que usar la columna "nacimiento":

```
mysql> SELECT * FROM mascotas WHERE nacimiento >= "2000-1-1";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| FanFan | Benito      | Perro   | m    | 2000-08-27 | NULL          |
| Wicho  | Tomás      | Ave     | NULL | 2000-02-09 | NULL          |
| Skim   | Benito     | Serpiente | m    | 2001-04-29 | NULL          |
| Pelusa | Diana      | Hamster  | f    | 2000-03-30 | NULL          |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Podemos también combinar condiciones, por ejemplo, para localizar a los perros hembras:

```
mysql> SELECT * FROM mascotas WHERE especie="Perro" AND sexo="f";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Buffy  | Arnoldo    | Perro   | f    | 1999-05-13 | NULL          |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

La consulta anterior usa el operador lógico AND. Hay también un operador lógico OR:

```
mysql> SELECT * FROM mascotas WHERE especie = "Ave" OR especie = "Gato";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Fluffy | Arnoldo    | Gato    | f    | 1999-02-04 | NULL          |
| Mau    | Juan       | Gato    | m    | 1998-03-17 | NULL          |
| Chispa | Omar       | Ave     | f    | 1998-09-11 | NULL          |
| Wicho  | Tomás     | Ave     | NULL | 2000-02-09 | NULL          |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

El operador AND y el operador OR pueden ser intercambiados. Si hacemos esto, es buena idea usar paréntesis para indicar como deben ser agrupadas las condiciones:

```
mysql> SELECT * FROM mascotas WHERE (especie = "Gato" AND sexo = "m")
-> OR (especie = "Perro" AND sexo = "f");
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Mau    | Juan       | Gato    | m    | 1998-03-17 | NULL          |
| Buffy  | Arnoldo    | Perro   | f    | 1999-05-13 | NULL          |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Seleccionando columnas particulares

Si no deseamos ver los registros completos de una tabla, entonces tenemos que usar los nombres de las columnas en las que estamos interesados separándolas por coma. Por ejemplo, si deseamos conocer la fecha de nacimiento de nuestras mascotas, debemos seleccionar la columna "nombre" y "nacimiento":

```
mysql> SELECT nombre, nacimiento FROM mascotas;
```

```

+-----+-----+
| nombre | nacimiento |
+-----+-----+
| Fluffy | 1999-02-04 |
| Mau    | 1998-03-17 |
| Buffy  | 1999-05-13 |
| FanFan | 2000-08-27 |
| Kaiser | 1989-08-31 |
| Chispa | 1998-09-11 |
| Wicho  | 2000-02-09 |
| Skim   | 2001-04-29 |
| Pelusa | 2000-03-30 |
+-----+-----+
9 rows in set (0.00 sec)

```

Para conocer quién tiene alguna mascota, usaremos la siguiente consulta:

```

mysql> SELECT propietario FROM mascotas;
+-----+
| propietario |
+-----+
| Arnoldo     |
| Juan        |
| Arnoldo     |
| Benito      |
| Diana       |
| Omar        |
| Tomás       |
| Benito      |
| Diana       |
+-----+
9 rows in set (0.00 sec)

```

Sin embargo, debemos notar que la consulta recupera el nombre del propietario de cada mascota, y algunos de ellos aparecen más de una vez. Para minimizar la salida, agregaremos la palabra clave **DISTINCT**:

```

mysql> SELECT DISTINCT propietario FROM mascotas;
+-----+
| propietario |
+-----+
| Arnoldo     |
| Juan        |
| Benito      |
| Diana       |
| Omar        |
| Tomás       |
+-----+
6 rows in set (0.03 sec)

```

Se puede usar también una cláusula **WHERE** para combinar selección de filas con selección de columnas. Por ejemplo, para obtener la fecha de nacimiento de los perritos y los gatitos, usaremos la siguiente consulta:

```

mysql> SELECT nombre, especie, nacimiento FROM mascotas
-> WHERE especie = "perro" OR especie = "gato";
+-----+-----+-----+
| nombre | especie | nacimiento |
+-----+-----+-----+
| Fluffy | Gato    | 1999-02-04 |
| Mau    | Gato    | 1998-03-17 |
| Buffy  | Perro   | 1999-05-13 |
| FanFan | Perro   | 2000-08-27 |
| Kaiser | Perro   | 1989-08-31 |
+-----+-----+-----+
5 rows in set (0.00 sec)

```

Ordenando registros

Se debe notar en los ejemplos anteriores que las filas regresadas son mostradas sin ningún orden en particular. Sin embargo, frecuentemente es más fácil examinar la salida de una

consulta cuando las filas son ordenadas en alguna forma útil. Para ordenar los resultados, tenemos que usar una cláusula ORDER BY.

Aquí aparecen algunos datos ordenados por fecha de nacimiento:

```
mysql> SELECT nombre, nacimiento FROM mascotas ORDER BY nacimiento;
+-----+-----+
| nombre | nacimiento |
+-----+-----+
| Kaiser | 1989-08-31 |
| Mau    | 1998-03-17 |
| Chispa | 1998-09-11 |
| Fluffy | 1999-02-04 |
| Buffy  | 1999-05-13 |
| Wicho  | 2000-02-09 |
| Pelusa | 2000-03-30 |
| FanFan | 2000-08-27 |
| Skim   | 2001-04-29 |
+-----+-----+
9 rows in set (0.00 sec)
```

En las columnas de tipo carácter, el ordenamiento es ejecutado normalmente de forma no sensible, es decir, no hay diferencia entre mayúsculas y minúsculas. Sin embargo, se puede forzar un ordenamiento sensible al usar el operador BINARY.

Para ordenar en orden inverso, debemos agregar la palabra clave DESC al nombre de la columna que estamos usando en el ordenamiento:

```
mysql> SELECT nombre, nacimiento FROM mascotas ORDER BY
-> nacimiento DESC;
+-----+-----+
| nombre | nacimiento |
+-----+-----+
| Skim   | 2001-04-29 |
| FanFan | 2000-08-27 |
| Pelusa | 2000-03-30 |
| Wicho  | 2000-02-09 |
| Buffy  | 1999-05-13 |
| Fluffy | 1999-02-04 |
| Chispa | 1998-09-11 |
| Mau    | 1998-03-17 |
| Kaiser | 1989-08-31 |
+-----+-----+
9 rows in set (0.00 sec)
```

Podemos ordenar múltiples columnas. Por ejemplo, para ordenar por tipo de animal, y poner al inicio los animalitos más pequeños de edad, usaremos la siguiente consulta:

```
mysql> SELECT nombre, especie, nacimiento FROM mascotas
-> ORDER BY especie, nacimiento DESC;
+-----+-----+-----+
| nombre | especie | nacimiento |
+-----+-----+-----+
| Wicho  | Ave     | 2000-02-09 |
| Chispa | Ave     | 1998-09-11 |
| Fluffy | Gato    | 1999-02-04 |
| Mau    | Gato    | 1998-03-17 |
| Pelusa | Hamster | 2000-03-30 |
| FanFan | Perro   | 2000-08-27 |
| Buffy  | Perro   | 1999-05-13 |
| Kaiser | Perro   | 1989-08-31 |
| Skim   | Serpiente | 2001-04-29 |
+-----+-----+-----+
9 rows in set (0.00 sec)
```

Notar que la palabra clave DESC aplica sólo a la columna nombrada que le precede.

Cálculos con fechas

MySQL proporciona diversas funciones que se pueden usar para efectuar cálculos sobre fechas, por ejemplo, para calcular edades o extraer partes de una fecha (día, mes, año, etc).

Para determinar la edad de cada una de nuestras mascotas, tenemos que calcular la diferencia de años de la fecha actual y la fecha de nacimiento, y entonces substrair uno si la fecha actual ocurre antes en el calendario que la fecha de nacimiento. Las siguientes consultas muestran la fecha actual, la fecha de nacimiento y la edad para cada mascota.

```
mysql> SELECT nombre, nacimiento, CURRENT_DATE,
-> (YEAR(CURRENT_DATE) - YEAR(nacimiento))
-> - (RIGHT(CURRENT_DATE,5) < RIGHT(nacimiento,5)) AS edad FROM mascotas;
+-----+-----+-----+-----+
| nombre | nacimiento | CURRENT_DATE | edad |
+-----+-----+-----+-----+
| Fluffy | 1999-02-04 | 2002-12-23 | 3 |
| Mau | 1998-03-17 | 2002-12-23 | 4 |
| Buffy | 1999-05-13 | 2002-12-23 | 3 |
| FanFan | 2000-08-27 | 2002-12-23 | 2 |
| Kaiser | 1989-08-31 | 2002-12-23 | 13 |
| Chispa | 1998-09-11 | 2002-12-23 | 4 |
| Wicho | 2000-02-09 | 2002-12-23 | 2 |
| Skim | 2001-04-29 | 2002-12-23 | 1 |
| Pelusa | 2000-03-30 | 2002-12-23 | 2 |
+-----+-----+-----+-----+
9 rows in set (0.01 sec)
```

Aquí, YEAR() obtiene únicamente el año y RIGHT() obtiene los cinco caracteres más a la derecha de cada una de las fechas, que representan el mes y el día (MM-DD). La parte de la expresión que compara los valores MM-DD se evalúa a 1 o 0, y permite ajustar el valor de la edad en el caso de que el valor MM-DD de la fecha actual ocurra antes del valor MM-DD de la fecha de nacimiento.

Dado que la expresión en sí es bastante fea, se ha usado un alias (edad) que es el que aparece como etiqueta en la columna que muestra el resultado de la consulta.

Esta consulta debe trabajar bien, pero el resultado puede ser de alguna manera más útil si las filas son presentadas en algún orden. Para ello haremos uso de la cláusula ORDER BY.

Por ejemplo, para ordenar por nombre, usaremos la siguiente consulta:

```
mysql> SELECT nombre, nacimiento, CURRENT_DATE,
-> (YEAR(CURRENT_DATE) - YEAR(nacimiento))
-> - (RIGHT(CURRENT_DATE,5) < RIGHT(nacimiento,5))
-> AS edad FROM mascotas ORDER BY nombre;
+-----+-----+-----+-----+
| nombre | nacimiento | CURRENT_DATE | edad |
+-----+-----+-----+-----+
| Buffy | 1999-05-13 | 2002-12-23 | 3 |
| Chispa | 1998-09-11 | 2002-12-23 | 4 |
| FanFan | 2000-08-27 | 2002-12-23 | 2 |
| Fluffy | 1999-02-04 | 2002-12-23 | 3 |
| Kaiser | 1989-08-31 | 2002-12-23 | 13 |
| Mau | 1998-03-17 | 2002-12-23 | 4 |
| Pelusa | 2000-03-30 | 2002-12-23 | 2 |
| Skim | 2001-04-29 | 2002-12-23 | 1 |
| Wicho | 2000-02-09 | 2002-12-23 | 2 |
+-----+-----+-----+-----+
9 rows in set (0.00 sec)
```

Para ordenar por edad en lugar de nombre, únicamente tenemos que usar una cláusula ORDER BY diferente:

```
mysql> SELECT nombre, nacimiento, CURRENT_DATE,
-> (YEAR(CURRENT_DATE) - YEAR(nacimiento))
-> - (RIGHT(CURRENT_DATE,5) < RIGHT(nacimiento,5))
-> AS edad FROM mascotas ORDER BY edad;
```

```

+-----+-----+-----+-----+
| nombre | nacimiento | CURRENT_DATE | edad |
+-----+-----+-----+-----+
| Skim   | 2001-04-29 | 2002-12-23   | 1   |
| FanFan | 2000-08-27 | 2002-12-23   | 2   |
| Wicho  | 2000-02-09 | 2002-12-23   | 2   |
| Pelusa | 2000-03-30 | 2002-12-23   | 2   |
| Fluffy | 1999-02-04 | 2002-12-23   | 3   |
| Buffy  | 1999-05-13 | 2002-12-23   | 3   |
| Mau    | 1998-03-17 | 2002-12-23   | 4   |
| Chispa | 1998-09-11 | 2002-12-23   | 4   |
| Kaiser | 1989-08-31 | 2002-12-23   | 13  |
+-----+-----+-----+-----+
9 rows in set (0.01 sec)

```

Una consulta similar puede ser usada para determinar la edad que tenía una mascota cuando falleció. Para determinar que animalitos ya fallecieron, la condición es que el valor en el campo fallecimiento no sea nulo (NULL). Entonces, para los registros con valor no-nulo, calculamos la diferencia entre los valores fallecimiento y nacimiento.

```

mysql> SELECT nombre, nacimiento, fallecimiento,
-> (YEAR(fallecimiento) - YEAR(nacimiento))
-> - (RIGHT(fallecimiento,5) < RIGHT(nacimiento,5))
-> AS edad FROM mascotas WHERE fallecimiento IS NOT NULL;
+-----+-----+-----+-----+
| nombre | nacimiento | fallecimiento | edad |
+-----+-----+-----+-----+
| Kaiser | 1989-08-31 | 1997-07-29   | 7   |
+-----+-----+-----+-----+
1 row in set (0.01 sec)

```

La consulta usa fallecimiento IS NOT NULL, en vez de fallecimiento < > NULL porque NULL es un valor especial. Esto será explicando más a detalle posteriormente.

¿Qué sucede si deseamos conocer cuáles de nuestras mascotas cumplen años el próximo mes? Para este tipo de cálculos, el año y el día son irrelevantes; simplemente tenemos que extraer el valor del mes en la columna nacimiento. Como se mencionó anteriormente, MySQL proporciona diversas funciones para trabajar y manipular fechas, en este caso haremos uso de la función MONTH(). Para ver como trabaja, vamos a ejecutar una consulta muy simple que muestra tanto el valor de una fecha como el valor que regresa la función MONTH().

```

mysql> SELECT nombre, nacimiento, MONTH(nacimiento) FROM mascotas;
+-----+-----+-----+
| nombre | nacimiento | MONTH(nacimiento) |
+-----+-----+-----+
| Fluffy | 1999-02-04 | 2   |
| Mau    | 1998-03-17 | 3   |
| Buffy  | 1999-05-13 | 5   |
| FanFan | 2000-08-27 | 8   |
| Kaiser | 1989-08-31 | 8   |
| Chispa | 1998-09-11 | 9   |
| Wicho  | 2000-02-09 | 2   |
| Skim   | 2001-04-29 | 4   |
| Pelusa | 2000-03-30 | 3   |
+-----+-----+-----+
9 rows in set (0.00 sec)

```

Encontrar los animalitos cuyo cumpleaños es el próximo mes es muy sencillo. Suponiendo que el mes actual es Abril (valor 4), entonces tenemos que buscar los registros cuyo valor de mes sea 5 (Mayo).

```

mysql> SELECT nombre, nacimiento FROM mascotas WHERE MONTH(nacimiento) = 5;
+-----+-----+
| nombre | nacimiento |
+-----+-----+
| Buffy  | 1999-05-13 |
+-----+-----+
1 row in set (0.00 sec)

```

Aquí habrá por supuesto una complicación si el mes actual es Diciembre. No podemos simplemente agregar uno al número del mes (12) y buscar los registros cuyo mes de nacimiento sea 13 porque dicho mes no existe. En vez de esto, tenemos que buscar los animalitos que nacieron en Enero (mes 1).

Sin embargo, lo mejor es que podemos escribir una consulta que funcione no importando cuál sea el mes actual. La función `DATE_ADD()` nos permite agregar un intervalo de tiempo a una fecha dada. Si agregamos un mes al valor regresado por la función `NOW()`, y entonces extraemos el valor del mes con la función `MONTH()`, el resultado es que siempre obtendremos el mes siguiente.

La consulta que resuelve nuestro problema queda así:

```
mysql> SELECT nombre, nacimiento FROM mascotas
-> WHERE MONTH(nacimiento) = MONTH(DATE_ADD(NOW(), INTERVAL 1 MONTH));
```

Trabajando con valores nulos

El valor `NULL` puede sorprendernos mientras no hayamos trabajado con él. Conceptualmente, `NULL` significa un valor que hace falta, o un valor desconocido, y es tratado de una manera diferente a otros valores. Para verificar si un valor es `NULL` no podemos usar los operadores de comparación tales como `=`, `>` o `<`.

Para probar esto ejecutemos la siguiente consulta:

```
mysql> SELECT 1 = NULL, 1 <> NULL, 1 < NULL, 1 > NULL;
+-----+
| 1 = NULL | 1 <> NULL | 1 < NULL | 1 > NULL |
+-----+
|      NULL |      NULL |      NULL |      NULL |
+-----+
1 row in set (0.00 sec)
```

Claramente observamos que no obtenemos resultados con algún significado con estos operadores. Es por ello que tenemos que usar los operadores `IS NULL` e `IS NOT NULL`:

```
mysql> SELECT 1 IS NULL, 1 IS NOT NULL;
+-----+
| 1 IS NULL | 1 IS NOT NULL |
+-----+
|          0 |          1 |
+-----+
1 row in set (0.00 sec)
```

En MySQL, 0 o `NULL` significan falso y cualquier otro valor significa verdadero. El valor que se considera verdadero por default es 1.

Cuando se usa un `ORDER BY`, los valores `NULL` son siempre ordenados primero, aún cuando se use la cláusula `DESC`.

Coincidencia de patrones

MySQL proporciona métodos de coincidencia de patrones basados en SQL estándar, así como también basados en expresiones regulares, de manera similar a las utilerías de Unix tales como `vi`, `grep` y `sed`.

La coincidencia de patrones basada en SQL nos permite usar `_` (guión bajo) para un solo carácter y `%` para un arbitrario número de caracteres. En MySQL, los patrones SQL no son sensibles al uso de mayúsculas y minúsculas.

Es importante notar que no se usan los operadores =, < o > cuando se usan los patrones SQL; en su lugar se usan los operadores LIKE y NOT LIKE.

A continuación se presentan algunos ejemplos.

Para encontrar los nombres que comienzan con b:

```
mysql> SELECT * FROM mascotas WHERE nombre LIKE "b%";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Buffy  | Arnoldo     | Perro   | f     | 1999-05-13  | NULL          |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Para encontrar los nombres que finalizan con fy:

```
mysql> SELECT * FROM mascotas WHERE nombre LIKE "%fy";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Fluffy | Arnoldo     | Gato    | f     | 1999-02-04  | NULL          |
| Buffy  | Arnoldo     | Perro   | f     | 1999-05-13  | NULL          |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Para encontrar nombres que contienen una s:

```
mysql> SELECT * FROM mascotas WHERE nombre LIKE "%s%";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Kaiser | Diana       | Perro   | m     | 1989-08-31  | 1997-07-29    |
| Chispa | Omar        | Ave     | f     | 1998-09-11  | NULL          |
| Skim   | Benito      | Serpiente | m     | 2001-04-29  | NULL          |
| Pelusa | Diana       | Hamster  | f     | 2000-03-30  | NULL          |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.01 sec)
```

El otro tipo de coincidencia de patrones proporcionado por MySQL hace uso de expresiones regulares. Para hacer uso de estos tipos de patrones se tienen que usar los operadores REGEXP y NOT REGEXP (o RLIKE y NOT RLIKE, los cuáles son sinónimos).

Algunas características de las expresiones regulares son:

- El caracter punto (.) coincide con cualquier caracter.
- Una clase de caracteres [...] coincide con cualquier caracter dentro de los paréntesis cuadrados. Por ejemplo, [abc] coincide con a, b o c. Para nombrar un rango de caracteres, se usa el guión. [a-z] coincide con cualquier letra minúscula, mientras que [0-9] coincide con cualquier dígito.
- El caracter asterisco (*) coincide con cero o más instancias de lo que le preceda. Por ejemplo, x* coincide con cualquier número de caracteres x, [0-9]* coincide con cualquier número de dígitos, y .* (punto asterisco) coincide con cualquier cosa.
- El patrón coincide si éste ocurre en cualquier parte del valor que está siendo evaluado. (Los patrones SQL coinciden únicamente en los valores completos.)
- Para indicar el inicio o el final de un valor que está siendo evaluado se usan los caracteres ^ y \$ respectivamente.

Para demostrar como se usan las expresiones regulares, se van a mostrar los ejemplos presentados anteriormente con el operador LIKE, ahora con el operador REGEXP.

Para encontrar los nombre que inician con b:

```
mysql> SELECT * FROM mascotas WHERE nombre REGEXP "^b";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Buffy | Arnoldo | Perro | f | 1999-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Antes de la versión 3.23.4 de MySQL, el operador REGEXP era sensible al uso de mayúsculas y minúsculas, así que dependiendo de la versión de MySQL con la que se está trabajando puede que obtengamos o no algún resultado en la consulta anterior. Se puede usar también la siguiente consulta para buscar los nombres que inician con la letra b, no importando si es mayúscula o minúscula.

```
mysql> SELECT * FROM mascotas WHERE nombre REGEXP "[bB]";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Buffy | Arnoldo | Perro | f | 1999-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.00 sec)
```

Desde la versión 3.23.4, para forzar que el operador REGEXP sea sensible al uso de mayúsculas y minúsculas, se tiene que usar la palabra clave BINARY para hacer de una de las cadenas, una cadena binaria. Observar los resultados de la siguientes consultas.

```
mysql> SELECT * FROM mascotas WHERE nombre REGEXP BINARY "^b";
Empty set (0.00 sec)

mysql> SELECT * FROM mascotas WHERE nombre REGEXP BINARY "^B";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Buffy | Arnoldo | Perro | f | 1999-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
1 row in set (0.01 sec)
```

Para encontrar los nombres que finalizan con la palabra fy, haremos uso del caracter \$.

```
mysql> SELECT * FROM mascotas WHERE nombre REGEXP "fy$";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Fluffy | Arnoldo | Gato | f | 1999-02-04 | NULL |
| Buffy | Arnoldo | Perro | f | 1999-05-13 | NULL |
+-----+-----+-----+-----+-----+-----+
2 rows in set (0.00 sec)
```

Para encontrar los nombres que contienen una letra s, la consulta sería:

```
mysql> SELECT * FROM mascotas WHERE nombre REGEXP "s";
+-----+-----+-----+-----+-----+-----+
| nombre | propietario | especie | sexo | nacimiento | fallecimiento |
+-----+-----+-----+-----+-----+-----+
| Kaiser | Diana | Perro | m | 1989-08-31 | 1997-07-29 |
| Chispa | Omar | Ave | f | 1998-09-11 | NULL |
| Skim | Benito | Serpiente | m | 2001-04-29 | NULL |
| Pelusa | Diana | Hamster | f | 2000-03-30 | NULL |
+-----+-----+-----+-----+-----+-----+
4 rows in set (0.00 sec)
```

Conteo de filas

Las bases de datos son usadas frecuentemente para responder una pregunta, "¿Con qué frecuencia ocurre un cierto tipo de dato en una tabla?". Por ejemplo, tal vez queremos conocer cuántas mascotas tenemos, o cuántas mascotas tiene cada uno de los propietarios.

Contar el número total de animalitos que tenemos es lo mismo que hacer la siguiente pregunta "¿Cuántas filas hay en la tabla mascotas?" ya que hay un registro por mascota. La función COUNT() es la que nos ayuda en esta situación.

```
mysql> SELECT COUNT(*) FROM mascotas;
+-----+
| COUNT(*) |
+-----+
|          9 |
+-----+
1 row in set (0.00 sec)
```

Si deseamos conocer cuántas mascotas tiene cada uno de los propietarios, la consulta es la siguiente:

```
mysql> SELECT propietario, COUNT(*) FROM mascotas GROUP BY propietario ;
+-----+-----+
| propietario | COUNT(*) |
+-----+-----+
| Arnoldo     | 2        |
| Benito      | 2        |
| Diana       | 2        |
| Juan        | 1        |
| Omar        | 1        |
| Tomás       | 1        |
+-----+-----+
6 rows in set (0.00 sec)
```

Se debe notar que se ha usado una cláusula GROUP BY para agrupar todos los registros de cada propietario. Si no hacemos esto, obtendremos un mensaje de error:

```
mysql> SELECT propietario, COUNT(*) FROM mascotas;
ERROR 1140: Mixing of GROUP columns (MIN(),MAX(),COUNT()...) with no
GROUP columns is illegal if there is no GROUP BY clause
```

En efecto, el uso de la función COUNT() en conjunto con la cláusula GROUP BY es muy útil en diversas situaciones. A continuación se muestran algunos ejemplos.

El número de animalitos por especie:

```
mysql> SELECT especie, COUNT(*) FROM mascotas GROUP BY especie ;
+-----+-----+
| especie | COUNT(*) |
+-----+-----+
| Ave     | 2        |
| Gato    | 2        |
| Hamster | 1        |
| Perro   | 3        |
| Serpiente | 1       |
+-----+-----+
5 rows in set (0.00 sec)
```

El número de animalitos por sexo:

```
mysql> SELECT sexo, COUNT(*) FROM mascotas GROUP BY sexo:
+-----+-----+
| sexo | COUNT(*) |
+-----+-----+
| NULL | 1        |
| f     | 4        |
| m     | 4        |
+-----+-----+
3 rows in set (0.01 sec)
```

El número de animalitos por combinación de especie y sexo:

```
mysql> SELECT especie, sexo, COUNT(*) FROM mascotas GROUP BY especie, sexo ;
+-----+-----+-----+
| especie | sexo | COUNT(*) |
+-----+-----+-----+
| Ave     | NULL | 1        |
| Ave     | f     | 1        |
| Gato    | f     | 1        |
```

```

| Gato      | m      | 1 |
| Hamster   | f      | 1 |
| Perro     | f      | 1 |
| Perro     | m      | 2 |
| Serpiente | m      | 1 |
+-----+
8 rows in set (0.00 sec)

```

No es necesario que se obtengan todos los datos de una tabla cuando se usa la función COUNT(). Por ejemplo, en la consulta anterior, para ver únicamente los datos de perritos y gatitos, la consulta queda de la siguiente manera:

```

mysql> SELECT especie, sexo, COUNT(*) FROM mascotas
-> WHERE especie="Perro" OR especie="Gato"
-> GROUP BY especie, sexo;
+-----+
| especie | sexo | COUNT(*) |
+-----+
| Gato    | f    | 1         |
| Gato    | m    | 1         |
| Perro   | f    | 1         |
| Perro   | m    | 2         |
+-----+
4 rows in set (0.00 sec)

```

O bien, si deseamos el número de animalitos por sexo, y cuyo sexo es conocido:

```

mysql> SELECT especie, sexo, COUNT(*) FROM mascotas
-> WHERE sexo IS NOT NULL
-> GROUP BY especie, sexo ;
+-----+
| especie | sexo | COUNT(*) |
+-----+
| Ave     | f    | 1         |
| Gato    | f    | 1         |
| Gato    | m    | 1         |
| Hamster | f    | 1         |
| Perro   | f    | 1         |
| Perro   | m    | 2         |
| Serpiente | m    | 1         |
+-----+
7 rows in set (0.00 sec)

```

Usando más de una tabla

La tabla mascotas nos ha servido hasta este momento para tener guardados los datos acerca de los animalitos que tenemos. Si deseamos guardar algún otro tipo de información acerca de ellos, tal como los eventos en sus vidas -visitas al veterinario, nacimientos de una camada, etc- necesitaremos de otra tabla. ¿Cómo deberá estar conformada esta tabla?. Lo que necesitamos es:

- El nombre de la mascota para saber a cuál de ellas se refiere el evento.
- Una fecha para saber cuando ocurrió el evento.
- Una descripción del evento.
- Un campo que indique el tipo de evento, si deseamos categorizarlos.

Dadas estas condiciones, la sentencia para crear la tabla eventos queda de la siguiente manera:

```

mysql> CREATE TABLE eventos(nombre varchar(20), fecha date,
-> tipo varchar(15), descripcion varchar(255));
Query OK, 0 rows affected (0.03 sec)

```

De manera similar a la tabla mascotas, es más fácil cargar los datos de los registros iniciales al crear un archivo de texto delimitado por tabuladores en el que se tenga la siguiente información: nombre fecha tipo descripción

```
Fluffy 2001-05-15 camada 4 gatitos, 3 hembras, 1 macho
Buffy 2001-06-23 camada 5 perritos, 2 hembras, 3 machos
Buffy 2002-06-19 camada 2 perritos, 1 hembra, 1 macho
Chispa 2000-03-21 veterinario Una pata lastimada
FanFan 2001-08-27 cumpleaños Primera vez que se enfermo de la panza
FanFan 2002-08-03 veterinario Dolor de panza
Whicho 2001-02-09 cumpleaños Remodelación de casa
```

Cargamos los datos con la siguiente sentencia:

```
mysql> LOAD DATA LOCAL INFILE "eventos.txt" INTO TABLE eventos;
Query OK, 7 rows affected (0.02 sec)
Records: 7 Deleted: 0 Skipped: 0 Warnings: 0
```

Tomando en cuenta lo que hemos aprendido en la ejecución de consultas sobre la tabla mascotas, debemos de ser capaces de recuperar algunos datos de la tabla eventos; los principios son los mismos. Sin embargo puede suceder que la tabla eventos por sí misma sea insuficiente para darnos las respuestas que necesitamos.

Supongamos que desemos conocer la edad de cada mascota cuando tuvieron una camada. La tabla eventos indica cuando ocurrió dicho evento, pero para calcular la edad de la madre, necesitamos sin duda su fecha de nacimiento. Dado que este dato está almacenado en la tabla mascotas, necesitamos de ambas tablas para realizar esta consulta.

```
mysql> SELECT mascotas.nombre,
-> (TO_DAYS(fecha) - TO_DAYS(nacimiento))/365 AS edad,
-> descripcion FROM mascotas, eventos
-> WHERE mascotas.nombre=eventos.nombre
-> AND tipo='camada';
+-----+-----+-----+
| nombre | edad | descripcion |
+-----+-----+-----+
| Fluffy | 2.28 | 4 gatitos, 3 hembras, 1 macho |
| Buffy | 2.12 | 5 perritos, 2 hembras, 3 machos |
| Buffy | 3.10 | 2 perritos, 1 hembra, 1 macho |
+-----+-----+-----+
3 rows in set (0.05 sec)
```

Hay diversas cosas que notar acerca de esta consulta:

- La cláusula FROM lista dos tablas dado que la consulta necesita información que se encuentra en ambas tablas.
- Cuando se combina (junta) información de múltiples tablas, es necesario especificar los registros de una tabla que pueden coincidir con los registros en la otra tabla. En nuestro caso, ambas columnas tienen una columna "nombre". La consulta usa la cláusula WHERE para obtener los registros cuyo valor en dicha columna es el mismo en ambas tablas.
- Dado que la columna "nombre" ocurre en ambas tablas, debemos de especificar a cuál de las columnas nos referimos. Esto se hace al anteponer el nombre de la tabla al nombre de la columna.

Nota: La función TO_DAYS() regresa el número de días transcurridos desde el año 0 hasta la fecha dada.

No es necesario que se tengan dos tablas diferentes para que se puedan juntar. Algunas veces es útil juntar una tabla consigo misma si se desean comparar registros de la misma

tabla. Por ejemplo, para encontrar las posibles parejas entre nuestras mascotas de acuerdo a la especie, la consulta sería la siguiente:

```
mysql> SELECT m1.nombre, m1.sexo, m2.nombre, m2.sexo, m1.especie
-> FROM mascotas AS m1, mascotas AS m2
-> WHERE m1.especie=m2.especie AND m1.sexo="f" AND m2.sexo="m";
+-----+-----+-----+-----+-----+
| nombre | sexo | nombre | sexo | especie |
+-----+-----+-----+-----+-----+
| Fluffy | f    | Mau    | m    | Gato    |
| Buffy  | f    | FanFan | m    | Perro   |
| Buffy  | f    | Kaiser | m    | Perro   |
+-----+-----+-----+-----+-----+
3 rows in set (0.00 sec)
```

En esta consulta se ha especificado un alias para el nombre de la tabla, y es éste el que se utiliza para referirse a las columnas.

Usando mysql en modo batch

En todos los ejemplos mostrados anteriormente, hemos usado mysql de manera interactiva para ejecutar algunas consultas y ver los resultados. Sin embargo, es posible usar mysql en modo batch. Para hacer esto tenemos que poner los comandos que deseamos ejecutar dentro de un archivo, y entonces decirle a mysql que lea los comandos de dicho archivo:

```
shell> mysql < archivo-batch
```

Si se usa mysql de esta manera, se está creando un pequeño script, y posteriormente se está ejecutando dicho script. Al ejecutar las sentencias y comandos que se encuentran en el script, es posible que suceda algún error. Si se desea que se continúen ejecutando las demás sentencias, a pesar de que haya ocurrido un error, se tiene que usar la opción `--force`

```
shell> mysql --force < archivo-batch
Así mismo, es posible especificar los parámetros de conexión desde la línea de comandos. Por ejemplo:
shell> mysql -h casita -u blueman -p < archivo-batch
```

¿Por qué usar un script?

Aquí hay algunas cuantas razones:

- Si se ejecutan un cierto número de consultas frecuentemente (cada día, o cada semana), al hacer un script nos evitamos tener que volver a teclear cada una de las consultas.
- Se pueden generar nuevas consultas que sean similares a las existentes al copiar y editar estos scripts.
- Al escribir consultas de varias líneas, los scripts ayudan bastante para que no se tengan que escribir todas las líneas nuevamente si se comete algún error.
- Si se están ejecutando consultas que producen una gran cantidad de datos, es posible usar un paginador para examinar los resultados de una mejor manera.

```
shell> mysql < archivo-batch | less
```

- Se puede guardar la salida en un archivo para revisarla posteriormente.

```
shell> mysql < archivo-batch > salida-del-script.txt
```

- Se pueden distribuir los scripts a otras personas para que puedan ejecutar también nuestros comandos y sentencias.

- En algunas situaciones no se permite el uso interactivo de mysql. Por ejemplo cuando se ejecuta un cron. En este caso, es indispensable usar el modo batch.

Cabe mencionar que el formato de la salida es diferente (más conciso) cuando se ejecuta mysql en modo batch, que cuando se usa de manera interactiva.

Ver el siguiente ejemplo.

La consulta es: `SELECT DISTINCT especie FROM mascotas.`

Si se ejecuta en modo interactivo:

```
mysql> SELECT DISTINCT especie FROM mascotas;
+-----+
| especie |
+-----+
| Gato    |
| Perro   |
| Ave     |
| Serpiente |
+-----+
4 rows in set (0.00 sec)
```

Si se ejecuta en modo batch:

```
shell> mysql -h casita -u blueman -p < especies-distintas.sql
Enter password: *****
especie
Gato
Perro
Ave
Serpiente
```

Si se desea obtener la salida que proporciona el modo interactivo, se tiene que usar la opción `-t`.

```
shell> mysql -t -h casita -u blueman -p < especies-distintas.sql
Enter password: *****
+-----+
| especie |
+-----+
| Gato    |
| Perro   |
| Ave     |
| Serpiente |
+-----+
```

2.6 Aplicaciones integrales APACHE + MYSQL + PHP

Hasta este punto se ha descrito la instalación de los tres servidores Apache, PHP y MySQL. Sin embargo, también existen aplicaciones en las que se instala un solo archivo que contiene los tres servidores. Se sugiere el uso de estos programas como segunda opción, sobre todo en el caso en que la instalación por separado no logra corregirse por algún detalle de configuración. Es mucho más sencilla la instalación de este tipo de aplicaciones por lo que se deja al estudiante su búsqueda e instalación.

Unidad 3. Scripts en PHP

PHP es un lenguaje de programación que se ejecuta en los servidores web y que permite crear contenido dinámico en tus páginas HTML. Dispone de miles herramientas que permiten acceder a bases de datos de forma sencilla, por lo que es ideal para crear aplicaciones para Internet.

En esta unidad se agrega el dinamismo a las páginas estáticas creadas con HTML las cuales se vuelven dinámicas gracias a php y mysql.

3.1 Fundamentos de PHP

PHP fue creado por Rasmus Lerdorf a finales de 1994, aunque no hubo una versión utilizable por otros usuarios hasta principios de 1995. Esta primera versión se llamó, *Personal Home Page* Tools [5].

Al principio, PHP sólo estaba compuesto por algunas macros que facilitaban el trabajo a la hora de crear una página Web. Hacia mediados de 1995 se creó el analizador sintáctico y se llamó PHP/F1 Versión 2, y sólo reconocía el texto HTML y algunas directivas de mSQL. A partir de este momento, la contribución al código fue pública.

El crecimiento de PHP desde entonces ha sido exponencial, y han surgido versiones nuevas como la actual, PHP5.

PHP es multiplataforma, funciona tanto para Unix (con Apache) como para Windows (con Microsoft Internet Information Server) de forma que el código que se crea para una de ellas no tiene porque codificarse al pasar a la otra. La sintaxis que utiliza, la toma de otros lenguajes muy extendidos como C y Perl, por lo que si se está familiarizado con estos, con PHP será fácil crear aplicaciones [1].

El funcionamiento es bastante simple:

- Escribes tus páginas HTML pero con el código PHP adentro.
- Guardas la página en el servidor web
- Un navegador solicita una página al servidor
- El servidor interpreta el código PHP
- El servidor envía resultado del conjunto de código HTML y el resultado del código PHP que también es HTML

En ninguno se envía código PHP al navegador, por lo que todas las operaciones realizadas son transparentes para el usuario, al que le parece que está visitando una página HTML que cualquier navegador puede interpretar [2][5].

Inserción de PHP en HTML

<? ?> Sólo si se activa la función **short_tags()** o la bandera de configuración *short_open_tag*.

<?php ?>

<script lenguaje="php"> </script>

<% %> Sólo si se activan los tags para ficheros '*asp*' con la bandera de configuración *asp_tags*.

Separación de instrucciones

Las instrucciones se separan con ';', en el caso de ser la última instrucción no es necesario el punto y coma.

Comentarios

Los comentarios en PHP pueden ser:

Como en C o C++, /*...*/ ó //

Otro tipo de comentario de una línea es #, que comentará la línea en la que aparezca pero sólo hasta el tag ?> que cierra el código php.

3.2 Declaración de variables y constantes

Los conceptos a tener en cuenta en PHP con las variables son los siguientes [5]:

- Cualquier nombre de variable está precedido por el símbolo \$.
- En PHP las variables siempre se asignan por valor, aunque en también existen métodos para asignaciones por referencia (&).

En PHP cada vez que se ejecuta un script, existen variables que se crean y que nos pueden informar del entorno en el que se está ejecutando dicho script. Para obtener una lista de todas estas variables predefinidas se puede utilizar la función *phpinfo()*.

De todas estas variables, algunas se crean dependiendo del servidor que se esté utilizando y otras son propias de PHP.

El ámbito de una variable en PHP es exactamente igual que en C o en Perl tomando siempre en cuenta los ficheros incluidos al principio de cada programa.

PHP permite un mecanismo para mantener variables con un nombre no fijo.

Por ejemplo:

```
$a = "hola";  
$$a = "mundo";
```

El ejemplo anterior, define dos variables, una denominada \$a que contiene el valor "hola" y otra que se llama \$hola que contiene el valor "mundo"

Para acceder al valor de una variable, se accede con:

```
echo "$a ${$a}";
```

ó

```
echo "$a ${$a}";
```

Ambas sentencias provocaran la salida "hola mundo".

Algo que se debe tener en cuenta cuando se utilizan variables, es que hay que resolver la ambigüedad que se crea al utilizar arrays de variables de este tipo. Por ejemplo \$\$a[1] provoca una ambigüedad para el intérprete, puesto que no sabe si se desea utilizar la variable denominada \$a[1] o utilizar la variables \$a indexandola en su primer valor. Para esto se utiliza una sintaxis especial que sería \${\$a[1]} o \${\$a}[1] según se desee una opción u otra.

Los tipos de cada variable en PHP no están tan claros como en C. El intérprete asigna el tipo de una variable según el uso que se esté haciendo de ella. Para asignar un tipo fijo a una variable se utiliza la función settype(). Los tipos son:

- Enteros
- Flotantes
- String
- Arrays
- Objetos
- Juggling

Respecto al tipo entero y flotante, no hay mucho que decir, así que detallaremos sólo los tipos String, Arrays, Objetos y Juggling.

String

Las cadenas pueden estar delimitadas por " o '. Si la cadena está delimitada por comillas dobles, cualquier variable incluida dentro de ella será sustituida por su valor. Para especificar el carácter " se escapará con el carácter backslash. Otra forma de delimitar una cadena es utilizando la sintaxis de documentos "<<<" Ejemplo:

```
$variable = <<< EOD
Ejemplo de cadena
que ocupa
varias líneas
EOD;
```

Esta última sintaxis sólo se puede utilizar con PHP 4. Las operaciones con cadenas son exactamente igual que en PERL.

Los arreglos en PHP se pueden utilizar tanto como Arrays indexados o como Arrays asociativos. Los Arrays de una sola dirección, pueden ser tanto escalares como asociativos. En realidad no existen ninguna diferencia entre ellos. Las funciones que se utilizan para crear Arrays de este tipo son list() o array() . En el caso de que no se especifique el índice en un array, el elemento que se asigna se añade al final.

Ejemplo:

```
$a[]="hola"
```

La instrucción anterior añade el string hola al final del array 'a'. Los arrays pueden ser ordenados utilizando las siguientes funciones: asort(), arsort(), ksort(), rsort(), sort(), uasort(), usort() y uksort() .

Otras funciones para el manejo de arrays son: count(), next(), prev() y each() .

En PHP, los arrays multidimensionales combinan las propiedades de un array unidimensional explicados anteriormente. Los índices de un array multidimensional pueden ser tanto numéricos como asociativos.

(Nota: hay que tener cuidado con la sintaxis de los arrays multidimensionales asociativos incluidos dentro de una cadena).

Ejemplo de array multidimensional asociativo:

```
$a=array(
    "manzana" => array("color" => "rojo", "tacto" => "suave"),
    "naranja" => array("color" => "naranja", "tacto" => "rugoso"),
    "platano" => array("color" => "amarillo", "tacto" => "suave")
);
```

Para inicializar un objeto se utiliza el método `new` , y para acceder a cada uno de sus métodos se utiliza el operador `->` .

Una variable en PHP, define su tipo según el contenido y el contexto en el que se utilice, es decir, si se asigna una cadena a una variable, el tipo de esa variable será `string` . Si a esa misma variable se el asigna un número, el tipo cambiará a `entero` . Para asegurarte de que una variable es del tipo adecuado se utiliza la función `settype()` . Para obtener el tipo de una variable se utiliza la función `gettype()` . También es posible utilizar el mecanismo del casting tal y como se utiliza en C.

Cuando existe un form en HTML, inmediatamente después de ser enviado, dentro del ámbito PHP se crean automáticamente una variable por cada uno de los objetos que contiene el form.

Si se activa la directiva `<?php_track_vars?>` o con la variable `track_vars` todo lo enviado por los métodos `POST` y `GET` estará en las variables `$HTTP_POST_VARS` y `$HTTP_GET_VARS`.

En PHP una expresión es cualquier cosa que pueda contener un valor. Las expresiones más simples son las variables y las constantes y otras más complicadas serán las funciones, puesto que cada función devuelve un valor al ser invocada, es decir, contiene un valor, por lo tanto, es una expresión.

Todas las expresiones en PHP son exactamente igual que en C. Los operadores abreviados, los incrementos, etc, son exactamente iguales. Incluso existen otros operadores adicionales como el operador `."` que concatena valores de variables, o el operador `"=="` denominado operador de identidad que devolverá verdadero si las expresiones a ambos lados del operador contienen el mismo valor y a la vez son del mismo tipo. Por último, el operador `"@"` sirve para el control de errores. Para poder ver como funciona el operador `@`, veamos un ejemplo:

```
<?php
$res = @mysql_query("select nombre from clientes")
or die ("Error en la selección, '$php_errormsg'");
?>
```

Este ejemplo, utiliza el operador `@` en la llamada a `mysql_query` y en el caso de dar un error, se salvará el mensaje devuelto en una variable denominada `php_errormsg`. Esta variable contendrá el mensaje de error de cada sentencia y si ocurre otro error posterior, se machaca el valor con la nueva cadena.

PHP mantiene también los operadores `""` que sirven para ejecutar un comando del sistema tal y como hace la función `system()` por ejemplo.

Las diferencias con C son los operadores de referencia, `&` y `*`, puesto que las operaciones por referencias no existen en PHP, aunque si son posibles en PHP4, y que en PHP existen

dos operadores and y dos operadores or que son: 'and', '&&' y 'or', '||' respectivamente, que se diferencian en la precedencia de cada uno.

La tabla que nos puede resumir la precedencia de cada uno de los operadores es:

Asocitividad	Operadores
Izquierda	,
Izquierda	or
Izquierda	xor
Izquierda	and
Derecha	print
Izquierda	= += -= *= /= .= %= &= = ^= ~= <<= >>=
Izquierda	?:
Izquierda	
Izquierda	&&
No posee	== != ===
No posee	< <= > >=
Izquierda	>> <<
Izquierda	+ - .
Izquierda	* / %
Derecha	! ~ ++ -- (int) (double) (string) (array) (Object) @
Derecha	[
No posee	new

Las estructuras de control en PHP se listan en la siguiente tabla:

Estructura	Alternativa
If, if else, if elseif	if: endif;
while	while: endwhile;
for	for: endfor;
do.. while	-
foreach(array as \$value) foreach(array as \$key=>\$value) (PHP4 y no PHP3)	-
switch	switch: endswitch;
continue	-
break	-

require()(Necesitan estar dentro de tags PHP)	-
include()(Necesitan estar dentro de tags PHP)	-

Una nota sobre require() y include(), Si se desea incluir un fichero de forma condicional, es mejor utilizar include(), sin embargo, si la línea donde está una instrucción require() no se ejecuta, no se ejecutará nada de ese fichero. Además, si en un bucle se ejecutan varias veces una instrucción require(), el intérprete lo incluirá una sólo vez, sin embargo si es include(), se incluirá el fichero cada vez que se ejecute la instrucción. Como apunte final, debes saber que en un fichero que va a ser requerido, se puede incluir una instrucción return al final como si esta instrucción devolviera un valor (sólo en PHP3), si se trata de include, se puede poner al final del fichero una instrucción return tanto en PHP3 como en PHP4, aunque con algunas diferencias.

Así, require, reemplaza su llamada por el contenido del fichero que requiere, e include, incluye y evalúa el fichero especificado.

3.3 ***Declaración de funciones***

Una de las herramientas mas importantes en cualquier lenguaje de programación son las funciones. Una función consiste en un conjunto de rutinas y acciones que a lo largo del script van a ser ejecutadas multitud de veces agrupados en una FUNCION y desde cualquier punto del script puede ser llamada y ejecutada. A su vez, esta función puede recibir parámetros externos de los cuales dependa el resultado de una función.

Las funciones deben ser colocadas siempre antes de realizar la llamada a la función (como es lógico). La sintaxis de una función es la siguiente:

```
function nombre(parámetros){
    instrucciones de la función
}
```

para llamar a la función sería de la siguiente forma: nombre(parámetros)

Un ejemplo es el siguiente. Crearemos una función que realice la suma de dos números y muestre el resultado

```
function sumar($sumando1,$sumando2){
    $ suma=$sumando1+$sumando2
    echo $sumando1."+".$sumando2."=".$suma;
}
```

sumar(5,6)

Un hecho relevante que cabe destacar es que las variables que declaremos dentro de la función solo existirán o tendrán dicho valor dentro de la función.

Existen casos en los cuales no sabemos el número de parámetros que le pasaremos a la función y en estos casos debemos usar las funciones creadas al efecto como son:

func_num_args() Numero de parámetros que se le han pasado a la función

func_get_args() Devuelve un elemento de los que forman la lista de argumentos

Un ejemplo de funciones definidas por el usuario puede ser:

```
function foo($arg1, $arg2, ..., $argN)
{
    echo "Función ejemplo"
    return $value;
}
```

Dentro de una función puede aparecer cualquier cosa, incluso otra función o definiciones de clase. Respecto al paso de argumentos, son siempre pasados por valor y para pasarlos por referencia hay que indicarlo y se puede hacer de dos formas diferentes, en la definición de la función, anteponiendo el símbolo & al argumento que corresponda, en este caso la llamada será igual que la llamada a una función normal, o manteniendo la definición de la función normal y anteponer un & delante del argumento que corresponda en la llamada a la función.

PHP permite el mecanismo de argumentos por defecto. Un ejemplo de esta característica es:

```
function hacerCafe($tipo="capuchino")
{
    return "he hecho un café $tipo\n";
}
```

En la llamada a esta función se obtendrá una frase u otra según se llame:

```
echo hacerCafe();
                                o
echo hacerCafe("expreso");
```

En el caso de tratarse de una función con argumentos por defecto y argumentos normales, los argumentos por defecto deberán estar agrupados al final de la lista de argumentos.

En PHP4 el número de argumentos de una función definida por el usuario, puede ser variable, se utilizan las funciones func_num_args(), func_get_arg() y func_et_args().

A diferencia de C, PHP puede devolver cualquier número de valores, sólo hará falta recibir estos argumentos de la forma adecuada. Ejemplo:

```
function numeros()      {
    return array(0,1,2);
}
list ($cero, $uno, $dos) = numeros();
```

3.4 Acceso a datos enviados a través de una página Web

Los Formularios no forman parte de PHP, sino del lenguaje estándar de Internet, HTML. Vamos a dedicar en este capítulo algunas líneas al HTML, para entrar posteriormente a tratarlos con PHP.

Todo formulario comienza con la etiqueta **<FORM ACTION="lo_que_sea.php" METHOD="post/get">** . Con . Con ACTION indicamos el script que va procesar la información que recogemos en el formulario, mientras que METHOD nos indica si el usuario del formulario va a enviar datos (post) o recogerlos (get). La etiqueta **<FORM>** indica el final del formulario.

A partir de la etiqueta **<FORM>** vienen los campos de entrada de datos que pueden ser:

Cuadro de texto:

```
<input type="text" name="nombre" size="20" value="jose">
```

Cuadro de texto con barras de desplazamiento:

```
<textarea rows="5" name="descripcion" cols="20">Es de color
rojo</textarea>
```

Casilla de verificación:

```
<input type="checkbox" name="cambiar" value="ON">
```

Botón de opción:

```
<input type="radio" value="azul" checked name="color">
```

Menú desplegable:

```
<select size="1" name="dia">
<option selected value="lunes">lunes</option>
<option>martes</option>
<option value="miercoles">miércoles</option>
</select>
```


Boton de comando:

```
<input type="submit" value="enviar" name="enviar">
```

Campo oculto:

```
<input type="hidden" name="edad" value="55">
```

Este último tipo de campo resulta especialmente útil cuando queremos pasar datos ocultos en un formulario.

Como habrás observado todos los tipos de campo tienen un modificador llamado name, que no es otro que el nombre de la variable con la cual recogeremos los datos en el script indicado por el modificador ACTION de la etiqueta FORM FORM, con value establecemos un valor por defecto.

A continuación veamos un ejemplo, para lo cual crearemos un formulario en HTML como el que sigue y lo llamaremos **formulario.htm**:

```
<HTML>
<BODY>
<FORM METHOD="post" ACTION="mis_datos.php">
<input type="hidden" name="edad" value="55">
<p>Tu nombre <input type="text" name="nombre" size="30"
value="jose"></p>
<p>Tu sistema favorito
<select size="1" name="sistema">
<option selected value="Linux">Linux</option>
<option value="Unix">Unix</option>
<option value="Macintosh">Macintosh</option>
<option value="Windows">Windows</option>
</select></p>
<p>¿Te gusta el futbol ? <input type="checkbox" name="futbol"
value="ON"></p>
<p>¿Cual es tu sexo?</p>
<blockquote>
<p>Hombre<input type="radio" value="hombre" checked
name="sexo"></p>
<p>Mujer <input type="radio" name="sexo" value="mujer"></p>
</blockquote>
<p>Aficiones</p>
<p><textarea rows="5" name="aficiones"
cols="28"></textarea></p>
<p><input type="submit" value="Enviar datos" name="enviar">
<input type="reset" value="Restablecer" name="B2"></p>
</FORM>
```

```
</BODY>
<HTML>
```

Y ahora creemos el script PHP llamado desde el formulario **mis_datos.php** :

Todos los datos se encuentran en la variable `$_POST`, ya que el formulario está enviado por el método post.

```
<?PHP;
if (isset($_POST['enviar'])) {
echo "Hola <b>" . $_POST['nombre'] . "</b> que tal
estás<BR>n";
echo "Eres " . $_POST['sexo'] . "<BR>n";
echo "Tienes " . $_POST['edad'] . "<BR>n";
echo "Tu sistema favorito es " . $_POST['sistema'] . "<BR>n";
if (isset($_POST['futbol'])) {
echo "Te gusta el futbol <BR>n";
} else odigo" style="margin-left: 50">} else {
echo "NO te gusta el futbol <BR>n";
}
}
if ($_POST['aficiones'] != "") {
echo "Tus aficiones son: <BR>n";
echo nl2br($_POST['aficiones']);
} else {
echo "NO tienes aficiones <BR>n";
}
}
echo "<a href='formulario.htm'>VOLVER AL FORMULARIO</a>"
?>
```

Una vez rellenados los datos del formulario, pulsamos el botón **Enviar datos** , con lo que el campo **enviar** toma lo que su etiqueta value indica, es decir **enviar="Enviar datos"** . En nuestro script lo primero que evaluamos es que se haya enviado el formulario, y para ello nada mejor que comprobar que la variable `$enviar` no está vacía. Le ponemos el signo dólar delante a **enviar** , ponemos el signo dólar delante a **enviar** , ya que en PHP todas las variables se les refiere con este signo.

Hay que tener en cuenta que si fusionáramos el código de ambos ficheros, nos ahorraríamos uno, pero no también se puede hacer en dos como lo estamos haciendo. Si la variable `$enviar` está vacía, enviamos el formulario.

```
<?PHP;
if ($enviar) {
echo "Hola <b>" . $nombre . "</b> que tal estás<BR>n";
echo "Eres " . $sexo . "<BR>n";
echo "Tienes " . $edad . "<BR>n";
echo "Tu sistema favorito es " . $sistema . "<BR>n";
if ($futbol) {
```

```

echo "Te gusta el futbol <BR>n";
} else {
echo "NO te gusta el futbol <BR>n";
}
if ($aficiones != "") {
< stuot;)>
echo "Tus aficiones son: <BR>n";
echo nl2br($aficiones);
} else {
echo "NO tienes aficiones <BR>n";
}
echo "<a href='$PHP_SELF'>VOLVER AL FORMULARIO</a>"
} else {
<HTML>
<BODY>
<FORM METHOD="post" ACTION="<?PHP echo $PHP_SELF ?>">
<input type="hidden" name="edad" value="55">
<p>Tu nombre <input type="text" name="nombre" size="30"
nombre" size="30" value="jose"></p>
<p>Tu sistema favorito
<select size="1" name="sistema">
<option selected value="Linux">Linux</option>
<option value="Unix">Unix</option>
<option value="Macintosh">Macintosh</option>
<option value="Windows">Windows</option>
</select></p>
<p>¿Te gusta el futbol ? <input type="checkbox" name="futbol"
value="ON"></p>
<p>¿Cual es tu sexo?</p>
<blockquote>
<p>Hombre<input type="radio" value="hombre" checked
name="sexo"></p>
<p>="codigo" style="margin-left: 100"><p>Mujer <input
type="radio" name="sexo" value="mujer"></p>
</blockquote>
<p>Aficiones</p>
<p><textarea rows="5" name="aficiones"
cols="28"></textarea></p>
<p><input type="submit" value="Enviar datos" name="enviar">
<input type="reset" value="Restablecer" name="B2"></p>
</FORM>
</BODY>
</HTML>
<?PHP
} //fin IF
?>

```

La variable de entorno `$PHP_SELF` , es una variable de entorno que nos devuelve el nombre del script que estamos ejecutando. Y por último, hacer notar el uso de la función `nl2br()` , `nl2br()` , con la cuál sustituimos los retornos de carro del texto, los cuáles no reconocen los navegadores, por la etiqueta `
` .

3.5 Generación de contenido dinámico

En el contexto de Internet, y concretamente referido a la World Wide Web, se denomina contenido a los textos, imágenes, videos, ficheros descargables, etc. que forman parte de las páginas web. En general, se distinguen dos tipos de contenidos: estáticos y dinámicos.

El contenido estático es aquél que permanece invariable desde el momento en que su autor lo crea. Es decir, no depende de quién lo visualice ni en que momento lo haga. Por ejemplo, un aviso legal.

El contenido dinámico es aquél que se genera automáticamente en el momento que alguien solicita su visualización, por tanto, puede cambiar dependiendo de quién lo solicite o en que momento lo haga. Por ejemplo, una sección de noticias.

Con lo expuesto en las secciones anteriores de este capítulo las páginas web pueden estar formadas por contenido dinámico.

3.6 Acceso a archivos

Vamos a ver un caso especial, como descargar un archivo desde un formulario. Para ello utilizaremos una etiqueta `INPUT` de tipo `FILE` , soportada a partir de las versiones de los navegadores Netscape Navigator 2.0 e Internet Explorer 4.0.

El formulario debe usar el método `post` , y el atributo `post` , y el atributo `enctype` debe tener el valor `multipart/form-data` . Además al formulario debemos añadirle un campo oculto de nombre `MAX_FILE_SIZE` , al cuál le daremos el valor en bytes del tamaño máximo del archivo a descargar.

```
<FORM ENCTYPE="multipart/form-data" ACTION="7-3.php"
METHOD="post">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="100000">
<INPUT NAME="archivo" TYPE="file">
<INPUT TYPE="submit" VALUE="Descargar Archivo">
</FORM>
```

Cuando el formulario es enviado, PHP detectará automáticamente que se está descargando un archivo y lo colocará en un directorio temporal en el servidor. Dicho directorio será que

el que esté indicado en el archivo de configuración **php.ini** , o en su defecto en el directorio temporal del sistema.

Cuando PHP detecta que se está descargando un archivo crea varias variables con el prefijo del nombre del archivo pero con distintas terminaciones. La variable terminada en `$_FILES['archivo']['name']` contiene el nombre original del archivo, `$_FILES['archivo']['size']` contiene el tamaño en bytes de éste, y la variable `$_FILES['archivo']['type']` nos indicará el tipo de archivo si éste es ofrecido por el navegador.

Si el proceso de descarga no ha sido correcto la variable archivo tomará el valor none y `_size` será 0 , y si el proceso ha sido correcto, pero la variable `$_FILES['archivo']['size']` da 0 , quiere decir que el archivo a descarga supera el tamaño máximo indicado por `MAX_FILE_SIZE` .

Una vez descargado el archivo, lo primero que debemos hacer es moverlo a otro lugar, pues sino se hace nada con él, cuando acabe la ejecución de la página se borrará.

Veamos un ejemplo.

```
<HTML>
<BODY>
<?PHP
if (isset($_POST['enviar'])) {
if ($_FILES['archivo']['name'] != "" &&
$_FILES['archivo']['size'] != 0){
echo "Nombre: $archivo_name <BR>n";
echo "Tamaño: $archivo_size <BR>n";
echo "Tipo: $archivo_type <BR>n";
if (! move_uploaded_file ($_FILES['archivo']['tmp_name'],
"directorio/".$_FILES['archivo']['name'])) {
echo "<h2>No se ha podido copiar el archivo</h2>n";
}
} elseif ($_FILES['archivo']['name'] != "" &&
$_FILES['archivo']['size'] == 0) {
echo "<h2>Tamaño de archivo superado</h2>n";
} else {
echo "<h2>No ha escogido un archivo para descargar</h2>n";
}
echo "<HR>n";
}
?>
<FORM ENCTYPE="multipart/form-data" ACTION="<?php echo
$_SERVER['PHP_SELF']; ?>" METHOD="post">
<INPUT type="hidden" name="MAX_FILE_SIZE" value="100000">
<p><b>Archivo a descargar<b><br>
<INPUT type="file" name="archivo" size="35"></p>
<p><INPUT type="submit" name="enviar" value="Aceptar"></p>
```

```
</FORM>
</BODY>
</HTML>
```

PHP permite la utilización de archivos remotos para realizar algún tipo de lectura de ellos. En el caso de querer realizar algún tipo de escritura, se debe hacer a través de un servidor ftp. Un ejemplo de ambas cosas se muestra a continuación.

```
<?php
$file = fopen("http://www.php.net/", "r");
if (!$file)
{
    echo "<p>Unable to open remote file.\n";
    exit;
}
while (!feof($file))
{
    $line = fgets($file, 1024); /* sólo funciona si todo está en una línea */
    if (eregi("(.)", $line, $out))
    {
        $title = $out[1];
        break;
    }
}
fclose($file);
?>
```

Unidad 4. Creación de aplicaciones

En esta unidad se crea una aplicación web completa, desde la descripción completa para definir los alcances de la aplicación hasta la implementación de la misma, considerando aspectos de usabilidad en cuanto al diseño de la interfaz y la validación que requieren los formularios para que el usuario incluya correctamente los datos para poder continuar con el envío de información.

4.1 Definición de la aplicación

El primer paso al iniciar la creación de una aplicación, ya sea para ejecutarse en web o no, consiste en definir la aplicación que se creará.

La consecución de los objetivos perseguidos a través de la puesta a disposición del público de cualquier aplicación web está condicionada por la satisfacción del usuario final. Los factores o atributos de calidad de una aplicación o sitio web que influirán en dicha satisfacción podemos clasificarlos en aquellos relacionados con: la calidad y utilidad de los contenidos; la calidad del servicio y asistencia del proveedor; y la calidad del diseño de la aplicación.

La importancia del diseño de la aplicación se basa en que éste será el que modele la interacción entre usuario y aplicación, y por tanto posibilitará o no la consecución de los objetivos perseguidos por el usuario (encontrar información, comprar, comunicarse, aprender...).

En el caso de la aplicación que se desarrolla durante el curso se sugiere definir desde un principio lo siguiente:

- Objetivo del sitio web que se construirá
- Usuarios a quienes va dirigido
- Secciones que contendrá
- Esquema de colores y fuentes que se desea utilizar
- Información que se incluirá en el sitio

Lo anterior es con base en el diseño centrado en el usuario que permite construir sistemas usables, de cualquier tipo. Una vez que se ha definido lo anterior, es necesario diseñar la interfaz del sistema, lo cual se explica con detalle en la siguiente sección.

4.2 *Diseño de la interfaz*

En esta fase se especifica el aspecto visual del sitio web: composición de cada tipo de página, aspecto y comportamiento de los elementos de interacción y presentación de elementos multimedia.

Con el objetivo de evitar la sobrecarga informativa, en el diseño de cada interfaz se debe tener en cuenta el comportamiento del usuario en el barrido visual de la página, distribuyendo los elementos de información y navegación según su importancia en zonas de mayor o menor jerarquía visual; por ejemplo, las zonas superiores del interfaz poseen más jerarquía visual que las inferiores.

Además de la posición de cada elemento en la interfaz, existen otras técnicas para jerarquizar información como son: uso del tamaño y espacio ocupado por cada elemento para otorgarle importancia en la jerarquía visual, utilización del contraste de color para discriminar y distribuir información, uso de efectos tipográficos para enfatizar contenidos, rotura de la simetría y uso de efectos de relieve / profundidad para resaltar elementos, etc.

Además de evitar la sobrecarga informativa jerarquizando los contenidos mediante las técnicas descritas, para evitar la sobrecarga memorística se recomienda definir menús de navegación con un número de opciones reducido, normalmente no más de nueve diferentes.

Otro aspecto importante en el diseño visual del sitio es la accesibilidad. En el uso de colores, por ejemplo, se debe ofrecer suficiente contraste entre texto y fondo para no dificultar la lectura, e igualmente seleccionar combinaciones de colores teniendo siempre en cuenta las discapacidades visuales en la percepción del color que pudieran presentar nuestros usuarios. [18]

Al utilizar imágenes en el diseño, por motivos de accesibilidad y comprensibilidad, se debe cuidar su resolución y tamaño, así como en fotografías la no pérdida de significación o contexto por recorte o minimización excesiva de la imagen.

Desde una perspectiva más amplia del diseño visual del sitio es importante mantener una coherencia y estilo común entre todas las páginas, proporcionando una consistencia visual a todo el sitio. Para asegurar que esta coherencia se cumple, es útil elaborar un libro o guía de estilo que sirva de documento referencia para todo el equipo de desarrollo.

La escritura hipertextual se debe realizar de forma diferente a la tradicional. El nuevo medio y sus características obligan a ser concisos, precisos, creativos y estructurados a la hora de redactar. Debemos conocer a quién nos dirigimos y adaptar el lenguaje, tono y vocabulario utilizado al usuario objetivo.

Algunos consejos a seguir en el diseño y redacción de contenidos son [18]:

- *Seguir una estructura piramidal* : La parte más importante del mensaje, el núcleo, debe ir al principio.
- *Permitir una fácil exploración del contenido* : El lector en entornos Web, antes de empezar a leer, suele explorar visualmente el contenido para comprobar si le interesa.
- *Un párrafo = una idea* : Cada párrafo es un objeto informativo. Se deben transmitir ideas, mensajes... evitando párrafos vacíos o varios mensajes en un mismo párrafo.
- *Ser conciso y preciso* : Al lector no le gusta leer en pantalla.
- *Vocabulario y lenguaje* : Se debe utilizar el mismo lenguaje del usuario, no el de la empresa o institución. El vocabulario debe ser sencillo y fácilmente comprensible.
- *Tono* : Cuanto más familiar y cercano (sin llegar a ser irrespetuoso) sea el tono empleado, más fácil será que el lector preste atención.
- *Confianza* : La mejor forma de ganarse la confianza del lector es permitiéndole el diálogo, así como conocer cuanta más información posible acerca del autor.

4.3 Validación de los formularios

Una de las grandes aportaciones de JavaScript a la creación de interfaces web es la posibilidad de acceder al contenido de los campos de los formularios para realizar acciones sobre los valores introducidos por el usuario, modificarlos y, en última instancia, validarlos.

La validación de los datos de un formulario mediante scripts JavaScript no sustituye a la validación que debe realizarse, por motivos de seguridad, en la aplicación del servidor que recibe la información. Sin embargo, al añadir una validación local con JavaScript, la experiencia de usuario mejora notablemente, al no ser necesario enviar los datos al servidor y esperar su respuesta para obtener sólo un mensaje informando de la incorrección de la información suministrada. Resulta frustrante cumplimentar un formulario, pulsar el botón enviar, y esperar 30 o 40 segundos para saber que hemos introducido mal un campo [10].

JavaScript, desde sus comienzos, introdujo los mecanismos necesarios para validar campos de formulario. Estas son algunas de las validaciones típicas:

- Comprobar que se han suministrado todos los campos obligatorios
- Comprobar que el formato de un campo es el esperado (fechas, teléfonos, etc.)
- Comprobar la validez (sintáctica) de las direcciones de correo y URLs
- Comprobar que no se sobrepasa la longitud, número de líneas o tamaño de la entrada

La siguiente guía ayuda a entender el funcionamiento de la validación de formularios con JavaScript, y enlaza con numerosos ejemplos de nuestra sección de código que pueden ser utilizados de modelo para crear la validación deseada.

Para ver los eventos y códigos disponibles para realizar la validación de formularios, consulte la primera unidad de estas notas de curso.

4.4 Implementación de la aplicación

Unidad 5. Publicación en la Web

Una vez creada la aplicación web, es útil publicarla en la Web para que pueda ser accesible a los usuarios de Internet. En las siguientes secciones se describe el proceso de publicación y mantenimiento de un sistema en línea.

5.1 *Requerimientos básicos*

Hoy en día hay muchas formas de publicar tu sitio web, las cuales deberás evaluar de acuerdo a lo que deseas comunicar en Internet. Lo primero que se requiere para publicar el sitio web creado es lo siguiente:

- Una cuenta con algún proveedor de hosting (gratuito o comercial)
- Un nombre de dominio (opcional)

La elección del hosting se condice absolutamente con el carácter de nuestro proyecto. Si nuestro sitio ha de tener páginas dinámicas necesitaremos bases de datos, sino cualquier hosting que nos satisfaga en cuanto a velocidad, estabilidad y servicio será adecuado. También hemos de tener en cuenta otras cuestiones tales como cuantas direcciones de correo nos permite alojar, que cuota de transferencia de archivos y que soporte nos ofrece a la hora de implementar formularios y otros servicios tales como dominios alias, subdominios, estadísticas, etc. Si nuestro sitio se publicará en un servidor gratuito o patrocinado podemos hacer nuestra búsqueda en: buscahosting.com y allí decidir cuál es el más adecuado a utilizar.

Si nuestra publicación tendrá dominio, es decir un nombre único a través del cual se identifica nuestro sitio, la elección es tal vez complicada por la gran variedad de opciones disponibles. Es importante tener en cuenta que cuanto más corto y más sencillo sea el nombre, más chances tenemos que los internautas vuelvan a acceder al sitio sólo por recordarlo más fácil.

Existen muchas empresas y entes oficiales a través de las cuales registrar un dominio. En general el costo no es alto y se paga por períodos de un año o más. La elección del nombre y de la extensión (.com, .com.es, .es, .org, .net, .edu, .tv, etc) se debe planificar de acuerdo al contenido del sitio, las posibilidades de nombres aún disponibles y lo que queramos transmitir ya con el nombre mismo de la URL de nuestro sitio.

Una vez que el sitio se encuentra en línea, se sugiere promocionarlo para que el público lo visite, para esto se puede registrar el sitio web en diferentes buscadores. Antes de empezar el proceso de registro en buscadores es muy recomendable que las páginas hayan sido revisadas y modificadas, incluyendo todas aquellas ayudas que permitan una mejor clasificación de las mismas. De este modo, conseguiremos que nuestras páginas sean fácilmente localizables entre los resultados del buscador.

Existen unas reglas y recomendaciones básicas para conseguir este objetivo. Elegir bien el título, descripción o palabras clave son algunos ejemplos. Vamos a ver con detenimiento en este reportaje cada una de estas técnicas. No nos tenemos que olvidar tampoco que el uso excesivo de estas puede ser contraproducente ya que muchos motores de búsqueda interpretan que la página esta haciendo "trampa" para situarse entre los primeros puestos de manera injusta. Cuando nuestra página es catalogada como "spam page" o "página tramposa" (en adelante spam), es confinada a los últimos puestos en los resultados de manera automática. Hay que utilizar estas técnicas con moderación, como se puede ver, por regla general.

5.2 Recursos gratuitos

En Internet existen multitud de recursos gratuitos para diseñar páginas webs. Así podemos encontrar manuales, tutoriales, alojamientos gratuitos, servicios interactivos (foros, chats, formularios, libros de visitas) que nos pueden ayudar a realizar muy buenos diseños. Los tipos de recursos que se requieren son: de construcción del sitio (editores html, css y php, manejadores de bases de datos) y de publicación del sitio.

Para el diseño y elaboración de páginas web existen diversos editores. Un editor de páginas Web es una aplicación diseñada con el fin de facilitar la creación de documentos HTML o XHTML. Su complejidad puede variar desde la de un simple editor de texto plano (como el Notepad de Windows), entornos WYSIWYG "lo que ves es lo que obtienes" (en inglés, *What You See Is What You Get*), hasta editores WYSIWYM ("lo que ves es lo que quieres decir", en inglés: *What You See Is What You Mean*).

Entre los editores (gratuitos y comerciales) que existen se encuentran los siguientes: CoffeeCup HTML Editor, Dreamweaver o Microsoft Frontpage, KompoZer (antes llamado NVU), Mozilla Composer, Amaya, Stone's WebWriter, Araneae 4.5.2, HTML-kit, EditPad Lite, 1st Page 2000 v2.0, NoteTab Light, Arachnophilia 4.0, EZPad, NotesPad, UltraEdit, Quanta (Linux).

Asimismo, algunos editores de páginas php son: Zend Studio, Editplus, PHP coder, DzSoft PHP Editor, PHP Designer, nuSphere PHPEd, entre otros.

En contraste, el manejador de bases de datos por mayoría de usuarios es MySQL aunque también el SQL Server es ampliamente usado, la diferencia está en el costo del segundo.

5.3 Administración remota

Cuando los visualizadores Web todavía no habían hecho su aparición en Internet, FTP (File Transfer Protocol, o Protocolo de Transferencia de Archivo) ya era una de las formas más usuales mediante la cual los usuarios de Internet podían transferir archivos desde y hacia sus computadoras.

Básicamente, el servicio de FTP se realiza a través de un programa FTP alojado en un servidor -llamado FTP daemon- el cual se encarga de gestionar las transacciones que solicita el cliente. Cuando éste se conecta, el daemon le pide que ingrese su nombre de usuario y contraseña, y en caso correcto permite el inicio de las "conversaciones" que darán lugar a las transferencias de archivos, sean éstas de tipo "download" ("bajadas" desde el servidor al computador del cliente) o "upload" ("subidas" desde la máquina del cliente al servidor).

Para realizar la administración remota del sitio se sugiere emplear algún cliente FTP en vez del gestor que incluyen algunos proveedores de hosting. La razón está en la velocidad de transferencia y la facilidad en el uso de los clientes FTP. Algunos programas son: WS FTP, CuteFTP, Smart FTP, FTP Voyager, Absolute FTP, los cuales poseen una interfaz gráfica que resulta muy cómoda e intuitiva para este tipo de acción.

Por otro lado, algunos editores como el Dreamweaver CS3 incluyen un cliente FTP para realizar la publicación del sitio desde el mismo editor, únicamente se requiere configurar los datos del servidor de hosting.

5.4 *Mantenimiento de un sitio web*

Considerando por un lado la dinámica propia de los negocios y, por otro, la constante evolución de las tecnologías web, resulta difícil imaginar que un sitio web pueda permanecer sin cambios durante largos períodos de tiempo. Si usted es un usuario asiduo de Internet, entenderá perfectamente de qué estamos hablando. Al igual que los negocios, los sitios web deben ser muy dinámicos y responder oportunamente a las necesidades de los clientes.

Por la naturaleza del medio de Internet, ningún sitio está totalmente terminado en definitiva. "Como la vida misma, los sitios web siempre están en un estado permanente de flujo: cambiando, creciendo y evolucionando". Por lo tanto, es importante actualizar continuamente el contenido publicado en los sitios web, desde el catálogo de productos, la lista de precios o incluso la información puesta al alcance del público.

Bibliografía

1. Andy Harris, PHP/MySQL Programming for the Absolute Beginner, Learning Express, 2003.
2. Cabezas Granado, Luis Miguel. PHP 5. Anaya Multimedia. 2004.
3. Chris Lea, Mike Buzzard, Dilip Thomas, Jessey White-Cinis, PHP MySQL Website Programming: Problem - Design – Solution. Editorial Apress. 2002.
4. Dondo, Agustín. Como interactuar con una base de datos MySQL usando PHP. En http://www.programacion.com/php/articulo/php_mysql/ (última consulta el 7 de julio de 2008).
5. García Arenas, María Isabel. Curso Comercio Electrónico, 2ª Edición. Geneura Team. En <http://geneura.ugr.es/~maribel/php/> (última consulta el 7 de julio de 2008).
6. Gil Rubio, Javier; Yagüe Panadero, Agustín; Tejedor Cerbel, Jorge; Alonso Villaverde, Santiago. Creación de sitios web con PHP 5. McGraw-Hill. 2005.
7. López Quijado, José. Domine PHP y MySQL. Programación dinámica en el lado del servidor. Editorial Ra-ma. Madrid, 2007.
8. Luke Welling, Laura Thomson, PHP and MySQL Web Development, Second Edition, SAMS, 2003.
9. Naramore, Elizabeth; Glass, Michael K.; Scouarnec, Yann Le. Desarrollo Web Con Php, Apache Y Mysql. Ed. Anaya Multimedia. 2004.
10. Orós, Juan Carlos. Diseño de páginas web interactivas con JavaScript y CSS. 3ª ed. Alfaomega RA-MA. 2002.
11. Pérez, César. Macromedia Dreamweaver MX 2004, desarrollo de páginas web dinámicas con PHP y MySQL. Alfaomega RA-MA. 2004.
12. Powers, David. Desarrollo web dinamico con dreamweaver 8 y php (diseño y creatividad). Anaya Multimedia-Anaya Interactiva. 2003.
13. Sitio web Programacion PHP. <http://www.programacionphp.net/recursos-programas/programas-de-editores-de-php.html>. (última consulta el 7 de julio de 2008).
14. The PHP Group. Manual de MySQL. Disponible en <http://mx.php.net/manual/es/> (última consulta el 7 de julio de 2008).
15. Tim Converse, Joyce Park, PHP Bible, 2nd Edition, Willey Publishing, Inc. 2000.
16. Tobias Ratschiller, Till Gerken. Creación de aplicaciones Web con PHP 4. Editorial Prentice Hall, Madrid, 2001.
17. Vaswani, Vikram. How To Do Everything With Php And Mysql. Editorial McGraw-Hill. USA, 2005.
18. Yusef Hassan & Francisco J. Martín Fernández & Ghzala Iazza. Diseño Web Centrado en el Usuario: Usabilidad y Arquitectura de la Información [on line]. "Hiptertext.net", núm. 2, 2004. <<http://www.hiptertext.net>>. ISSN 1695-5498. (última consulta el 7 de julio de 2008).