



**TECNOLOGICO  
DE MONTERREY**

MÓDULO VII:  
DIPLOMADO EN DISEÑO Y CREACIÓN DE PÁGINAS WEB

**Una introducción práctica al uso de base de datos  
en aplicaciones web**

**Trabajando con PHP y MySQL**



Instructor:

M Cs. Javier González Sánchez  
javiergs@acm.org

**PACHUCA, HIDALGO  
31 DE OCTUBRE Y 1 DE NOVIEMBRE DE 2003**



## **OBJETIVO GENERAL:**

Al finalizar el modulo, el participante será capaz de crear páginas web que interactúen con una base de datos. Esto implica insertar, modificar y consultar información almacenada en un gestor de base de datos a través de páginas web. Obteniendo conocimiento y habilidades necesarias que le permitirán construir, diseñar, publicar y realizar un sitio web dinámico.

## **TEMARIO**

1. Cómo establecer la conexión con la base de datos
2. Cómo añadir un dato dinámico
3. Cómo crear una página con múltiples objetos
4. Crear páginas de inserción
5. Crear páginas de actualización
6. Crear variables de búsqueda con SQL
7. Mostrar objetos del servidor
8. Configuración de contraseñas y de la seguridad
9. Crear comportamientos del servidor

## Introducción

Para muchos la combinación de PHP y MySQL es la mejor opción para el desarrollo de sitios web dinámicos. Ambos son parte del mundo "open source" o software libre: gratuitos, con grupos de desarrollo independientes en todo el mundo.

En este modulo dejaremos de lado por un instante los elementos visuales y de diseño de un sitio web y trabajaremos tras bambalinas con los elementos propios de la información y lógica de negocio. Dividimos nuestro modulo en tres unidades:

En la primera unidad conocerás brevemente como instalar y configurar una base de datos. Además de cómo elaborar scripts que permitan introducir información en la base de datos y desplegar esta en un sitio web.

En la segunda unidad se cubre en mayor detalle el trabajo de programación con PHP en conjunto con la base de datos MySQL. Comenzamos comentando el manejo de ciclos y condicionales. Y veremos como PHP puede combinarse con formularios contruidos con HTML. Al término de esta unidad usted será capaz de añadir, editar y eliminar información de su base de datos.

En la tercera unidad mostraremos algunos de los secretos que convierten un sitio web con datos en una verdadera aplicación web. Se mostrara como revenir que los usuarios dejen campos en blanco, como asegurar que se proporcionen datos numéricos o alfabéticos según se requiera. Además de mostrar como PHP controla el reuso de código y el encapsulado de acciones en módulos.

## Unidad 1.

### Introducción a PHP y MySQL

Seguramente si usted ha estado en contacto con el mundo de la informática habrá escuchado el concepto "open source software". Este movimiento ha ganado importancia a tal grado que compañías como Oracle, Informix y otras mas están tomando ventaja de los productos generados como software "open source". Y que decir del sistema operativo en bogue: Linux.

Si contamos con un sistema masivamente complejo de RDBMS (sistema administrador de bases de datos relacionales) es tan bueno, tanto como si sabemos que hacer con él.

Nosotros solo daremos una introducción al mundo de las bases de datos. Tan profundo como para realizar un sitio web sustentado en información dinámica. Y si hemos de aprender ha hacer eso, que mejor que lograrlo de una manera en la que nuestro trabajo pueda ser implantado en cualquier equipo, tanto en servidores con Microsoft Windows en cualquiera de sus "sabores y colores" hasta equipos mas sofisticados con sistemas operativos Unix también en cualquiera de sus modelos.

Una encuesta de **Netcraft** ([www.netscraft.com](http://www.netscraft.com)) muestra que el uso de PHP creció de 7,500 sitios en junio de 1998 a 410,000 en marzo de 1999. Por su parte MySQL fue premiada como la base de datos (libre) del año en el Webcon de 1998.

MySQL es una base de datos pequeña y compacta ideal para aplicaciones pequeñas y medianas. Con soporte para el estándar SQL, corre en un sin numero de plataformas con un rendimiento a la altura de las mejores bases de datos comerciales.

PHP es un lenguaje para generación de "scripts" que han de ser colocados en un servidor de web para ser funcionales. PHP permite colocar código de programación incrustado en páginas HTML. Cuando el servidor de web encuentra elementos de PHP en una pagina HTML que ha sido solicitada por un navegador, envía los elementos HMTML intactos, pero el script PHP es ejecutado y los resultados de la ejecución del script son anexados al envío en lugar del código PHP original.

La combinación de PHP/MySQL es gratuita (sin restricción alguna sobre licencias) y es independiente de la plataforma. Sin embargo PHP también soporta otras plataformas de base de datos incluidos Informix, Oracle, Sybase, Solid, PostgreSQL, y claro cualquier otra que sea capaz de operar vía ODBC.

PHP tiene muchas otras cualidades, entre otras, permitir la autenticación de usuarios, el uso de XML, la creación dinámica de imágenes, creación dinámica de archivos PDF y muchas mas. Es además fácilmente extensible, pequeño y sencillo. Una maravilla desde el punto de vista de algunos desarrolladores.

Finalmente cabe mencionar que para ambos proyectos, PHP y MySQL existe una gran cantidad de información y documentación disponible en la web. Errores y problemas en ambos productos son solucionados con una rapidez considerable y nuevas características se implementan día con día.

## 1.1. Instalación de nuestras herramientas.

Realmente no necesitamos preocuparnos por el proceso de instalación la mayor parte de las veces dejaremos esta labor a nuestro proveedor de acceso a Internet, quien se encarga de hospedar nuestro sitio web. Sin embargo es adecuado conocer el proceso involucrado, sobretodo si deseamos realizar una instalación de estos productos en nuestra computadora personal para aprender a utilizarlos.

## 1.2. Instalación de la base de datos: MySQL

Comencemos por obtener el software que necesitamos para trabajar. Comencemos por la base de datos.

El sitio web oficial de MySQL se encuentra en [www.mysql.com](http://www.mysql.com). Será sencillo para usted ingresar a la sección etiquetada "downloads" para descargar el software que requerimos. Tomemos una versión estable, en este momento MySQL 4.0 es una buena opción. Aunque podríamos experimentar con cualquier plataforma, lo mas factible es que usted al igual que yo tenga a su alcance una computadora con Microsoft Windows para comenzar a trabajar, así que seleccionemos la versión **Windows 95/98/NT/2000/XP/2003** y tomemos un descanso mientras los 22.7 Mb del software se transfieren a nuestro equipo. Siendo MySQL un software de uso tan difundido, es de entenderse que exista mas de un nodo en Internet desde el cual podamos obtenerla, cuando el web se lo pregunte elija el más cercano a usted. Claro recuerde que la cercanía física no significa necesariamente una mayor rapidez de transferencia por la red. Si todo va bien tendrás en tu equipo un archivo llamado **mysql-4.0.16-win.zip** o alguno similar con una numeración distinta, según la versión descargada.

Descompacta el archivo y ejecuta setup.exe

Sigue las indicaciones en pantalla. Para nuestros fines la instalación básica de MySQL es suficiente, así que selecciona la opción "instalación típica" para evitar detalles y complicaciones. Para obtener mayor información de las diferentes opciones de instalación consulta la documentación de MySQL en el sitio web [www.mysql.com](http://www.mysql.com).

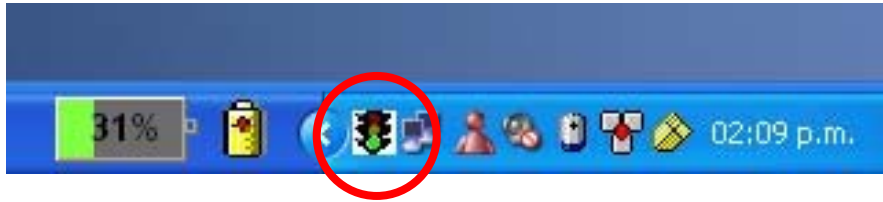
En este momento es buena idea reiniciar la computadora para que la instalación sea concluida satisfactoriamente.

Localiza el archivo winmysqladmin.exe en el subdirectorio bin bajo el directorio donde realizaste la instalación del producto. Por ejemplo

```
C:\mysql\bin\winmysqladmin.exe
```

Ejecuta el archivo. La primera ocasión que lo ejecutes te solicitara crear tu clave de administrador. Proporciona el que desees y procura no olvidarlo.

Si todo va bien en este momento veras sobre la barra de inicio de Windows un icono como esto:



Un semáforo con la luz verde, que nos indica podemos continuar con nuestra labor.

Estamos a la mitad del camino, ya contamos un sistema gestor de base de datos, ahora queremos instalar PHP.

### 1.3. Instalación del servidor web: Apache

Como comentamos en la introducción, PHP es un lenguaje que permite incrustar código de programación en hojas de web. A fin de poder operar PHP debe trabajar de la mano del servidor donde se hospedan las hojas web. Así que para poder elaborar scripts PHP y probarlos requerimos o bien contar con acceso a un servidor web que contratemos con un proveedor de servicios o bien instalar uno propio. Si bien para publicar nuestro sitio web oficial requeriremos de contratar un espacio con un proveedor de servicios de Internet, por el momento podemos instalar uno en nuestra computadora.

Sigamos tomando ventaja del software "open source" y ahora descarguemos un servidor de web o servidor de http. El elegido es el servidor web Apache. Apache es el servidor de web mas utilizado en Internet, estadísticas del sitio web de Netscraft colocan a Apache con un 66% de sitios en la red.

Ingresemos al sitio oficial de Apache en [www.apache.org](http://www.apache.org) y localicemos el enlace a **Httpd server Apache 2.0.47**

Para nuestros fines cualquiera de las versiones disponibles resulta de utilidad. Sin embargo es buena idea tomar la versión mas reciente disponible.

Descarguemos el archivo con el servidor compatible con windows, el archivo **apache\_2.0.47-win32-x86-no\_ssl.msi** con aproximadamente 5.80 Mb de tamaño.

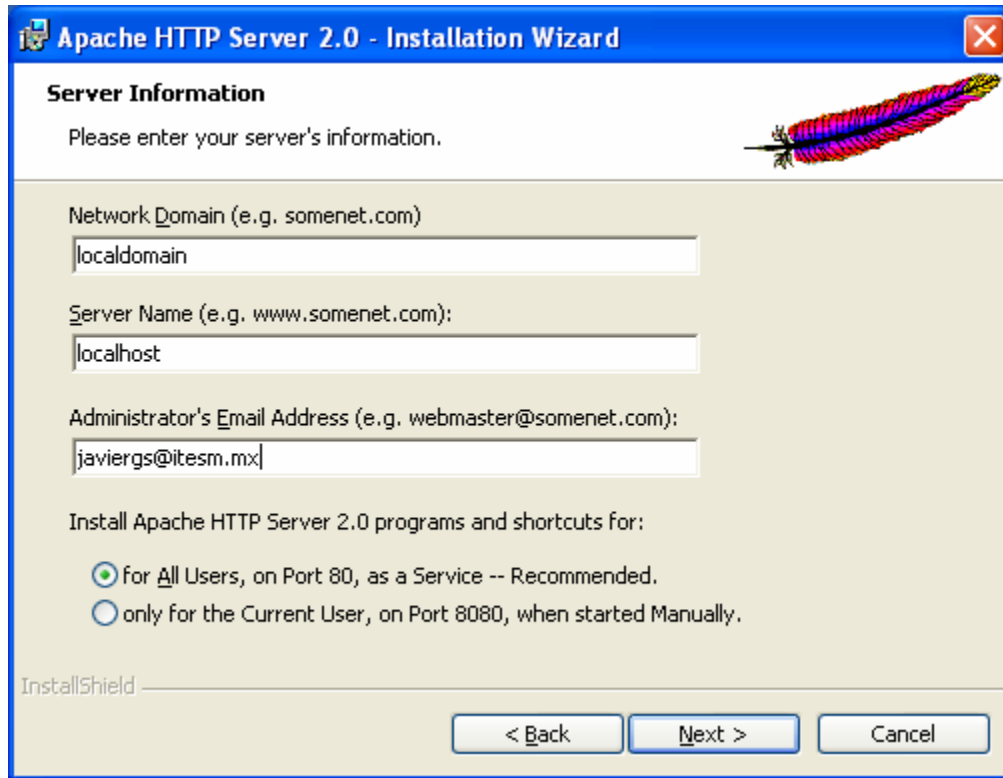
Una vez descargado el archivo, doble clic en el para comenzar con el proceso de instalación. Aceptemos la licencia y proporciona los siguientes datos en la ventana de configuración:

Dominio: localdomain

Nombre del Servidor: localhost

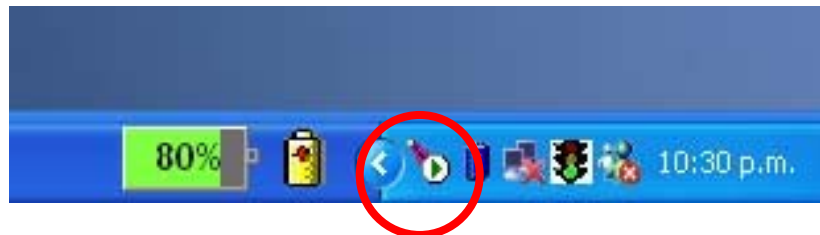
Correo del administrador: <escribe aquí tu correo electronico>

Apache es un software que por si mismo requeriría una explicación aparte. Dado que sale del alcance de este modulo, simplemente confía en la configuración que te propongo.



Una vez completa la pantalla mostrada clic en el botón seguir y cuando el proceso de instalación te lo pregunte elige la instalación "típica". Ello es suficiente para nuestras necesidades.

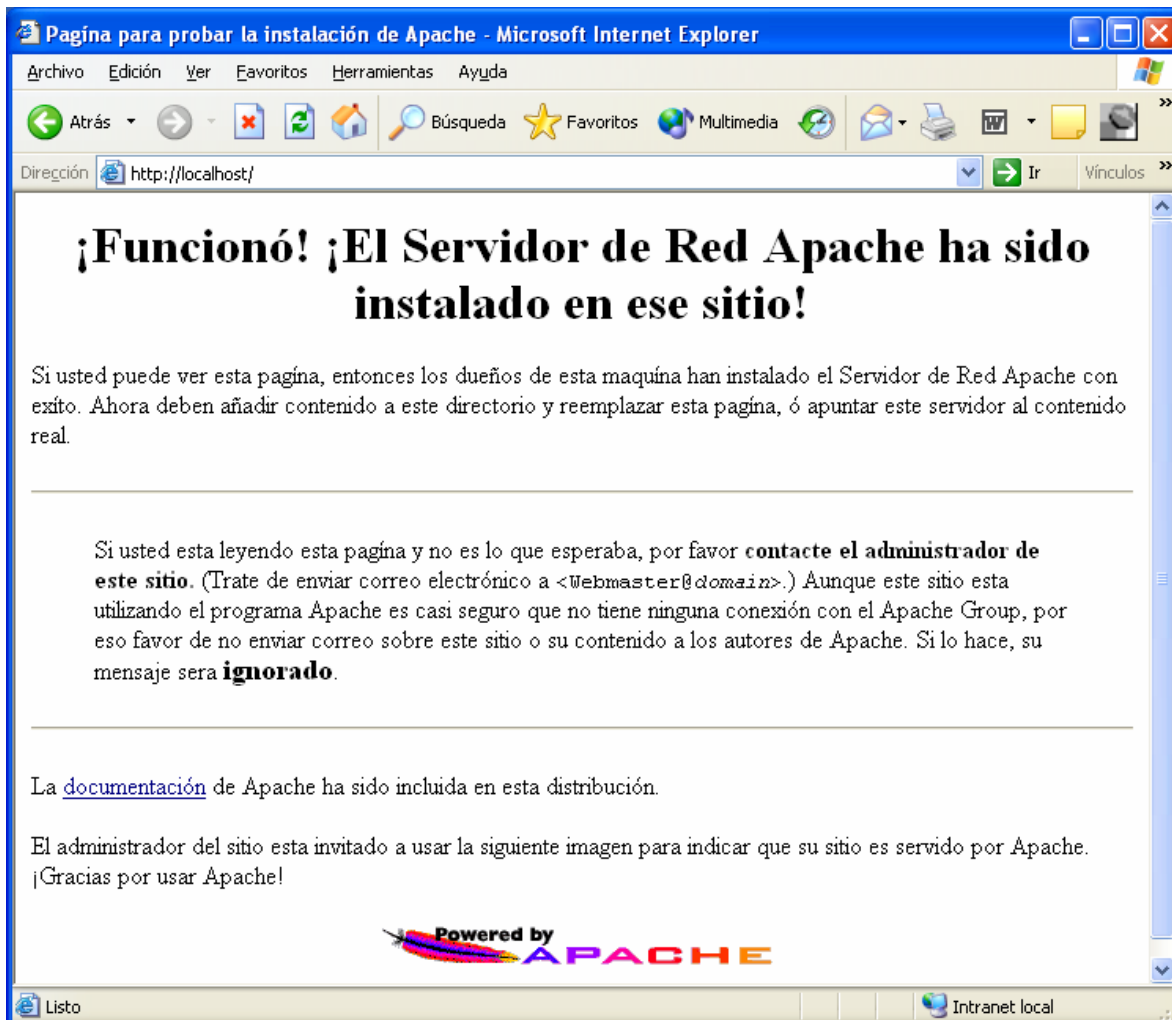
Concluida la instalación, revisemos nuestra barra de inicio en Windows. Un nuevo icono confirma que el proceso es exitoso y podemos continuar



Para confirmar abre un navegador en tu computador y conecta la siguiente dirección:

<http://localhost/>

La siguiente página aparecerá y nos invita a continuar el proceso



### 1.3. Instalación de PHP

Instalar el gestor de base de datos MySQL fue relativamente sencillo al igual que el servidor de web, veamos ahora el proceso a seguir para contar con PHP.

También parte de la familia "open source" PHP puede ser descargado del sitio web [www.php.net](http://www.php.net).

Descargaremos de ahí en particular de la sección Windows el paquete **php-4.3.3-Win32.zip**, aproximadamente 6.1Mb serán transferidos a nuestra computadora

Toma el paquete u descompactalo en una carpeta de tu computador. Te recomiendo coloques el fólder en el mismo sitio donde has estado instalando el resto de los paquetes. En mi equipo tengo en este momento las carpetas:

C:\mysql

C:\Apache2

C:\php

Te recomiendo descompactar php en la unidad c en un f3lder llamado php. Si decides utilizar una ruta distinta, recuerda cambiar "C:\php" por el directorio por ti utilizado en todos los pasos mencionados a continuaci3n como parte del proceso de instalaci3n.

El archivo install.txt ubicado en el c:\php explica el proceso de instalaci3n a detalle. Sin embargo comentaremos a continuaci3n los pasos a seguir.

En el directorio php se encuentra el archivo php4ts.dll copialo en el directorio C:\windows\system

Luego renombra el archivo php.ini-dist como php.ini y copialo en el directorio C:\windows

Hecho lo anterior solo nos resta conectar nuestra instalaci3n de PHP con la instalaci3n del servidor web Apache, de forma tal que nuestro servidor web pueda reconocer y ejecutar las instrucciones que se le dar3n empleando el lenguaje PHP.

Ser3 necesario que a3adamos las siguientes l3neas en el archivo httpd.conf que se encuentra en el f3lder c:\Apache2\conf

```
LoadModule php4_module D:/php/sapi/php4apache2.dll
# AddModule mod_php4.c
AddType application/x-httpd-php .php
```

Para que los cambios tomen efecto, ser3 necesario detener nuestro servidor de web y ejecutarlo nuevamente. En tu men3 de inicio encontraras bajo la opci3n de programas, las herramientas detener y ejecutar para Apache http Server.

Hasta aqu3 hemos terminado con la configuraci3n de nuestro ambiente de trabajo. Podemos tomar un peque3o receso.

Como para cualquier otro software la mayor3a de las veces no requerimos ser expertos en la instalaci3n de los productos para poderlos utilizar adecuadamente.

Rescapitulando:

En este momento tenemos operando en nuestra computadora un gestor de base de datos (MySQL), un servidor de web (Apache) y un lenguaje de programaci3n para web (PHP).

¡Ahora, pongamos eso a trabajar!

## 1. 4. Mi primer programa

Aqu3 es donde comienza nuestra labor. Ahora que todo el software esta instalado y configurado, probemos que todo funciona adecuadamente.

Escribe el siguiente bloque de texto en un archivo y almacena el archivo en el siguiente directorio:

C:\Apache2\htdocs\curso\

En un archivo llamado primero.php

```
<html>
<body>

<?php
    $myvar = "Hello World";
    echo $myvar;
?>

</body>
</html>
```

Ahora abre un navegador y escribe el siguiente URL para conectarte a tu servidor web y ejecutar tu primer programa

<http://localhost/curso/primer.php>

El navegador te despliega una pagina con el texto "Hello World".

Si en su lugar aparece cualquier tipo de mensaje de error, revisa la documentación de PHP para obtener información respecto a como corregir el error.

Si todo esta bien y solicitas a tu navegador te muestre el código fuente de la pagina desplegada, observarás que solo existe una línea en el código fuente

Hello World

Esto es debido a que PHP examino la página, proceso el código que escribimos originalmente y envió al navegador solo el código HTML que en el programa definimos, el resto son instrucciones que el usuario (la persona que ve nuestra página a través de un navegador web) no debe conocer.

La primera cosa que debiste notar del script de arriba, son los delimitadores. Hay líneas que indican el inicio y el fin del bloque. El poder de PHP, es que este puede ser introducido en cualquier lugar en tu código, de alguna manera. Mas tarde veremos algunos interesantes usos para esto, pero por ahora vamos a dejarlo así de simple. Si quieres, puedes configurar PHP para usar etiquetas cortas, pero no son compatibles con XML, se cuidadoso. Si estas haciendo un switch para ASP, puedes configurar el PHP para usar delimitadores <% y %>.

Otra cosa que debes notar es el punto y coma, al final de cada línea. Esto lo conoceremos como un separador y sirve para distinguir un grupo de instrucciones de otro. Es factible escribir un programa PHP en una línea y separarlo en porciones con puntos y comas. Pero seria confuso, así que adicionamos una nueva línea después de cada punto y coma. Solo recuerda que cada línea debe finalizar con un punto y coma.

Finalmente, observa que la palabra myvar inicia con un signo de dólar(\$). Este símbolo le dice a PHP que es una variable. Nosotros asignamos la palabra "Hola mundo" a la

variable `$myvar`. Una variable puede también contener números o un arreglo. De cualquier manera, todas las variables inician con un símbolo de dólar.

El poder Real de PHP viene de sus funciones. Estas son básicamente instrucciones procesadas. Existen más de 700 funciones disponibles que nos permiten realizar diversas actividades. Así que hay bastantes cositas que puedes hacer.

Ahora vamos a adicionarle MySQL a nuestro trabajo.

### 1.5. Crear una base de datos

Ahora estamos listos para conectarnos a MySQL. Una forma sencilla de conocer que opciones están disponibles en PHP y que hacen en tu servidor es usando la función `phpinfo()`. Crea un script con lo siguiente

```
<html>
<body>

<?php
    phpinfo();
?>

</body>
</html>
```

Salva y ve este script a través de tu servidor de Web. Observaras una página con información útil e interesante. Esa información dice todo acerca de tu servidor, el ambiente interno de las variables del servidor de Web, las opciones que son compiladas y mucho más.

Si MySQL esta ahí, entonces estas listo para continuar.

Antes que puedas obtener datos de MySQL, tienes que poner datos en ella. No hay una manera fácil de hacer esta etapa. La mayoría de los scripts de PHP viene con un archivo conocido como core dump que contiene todos los datos requeridos para crear y almacenar una base de datos MySQL. La inserción y salida de este proceso esta realmente fuera del objetivo de este texto, así que lo haremos sin explicar a detalle el proceso. Consulta información respecto al lenguaje SQL y sistemas de gestión de bases de datos en las fuentes necesarias.

MySQL utiliza su propia tabla de usuarios. En la instalación, un usuario default (root) es automáticamente creado sin password. Es el administrador de la base de datos, podrían adicionarse nuevos usuarios con varios permisos, pero se podría escribir un artículo entero sobre eso, así que trabajaremos como usuario root.

Para ingresar a la base de datos, se requiere algún trabajo en la consola de DOS. Se tendrá que utilizar una ventana DOS y ubicarnos en la ruta del directorio MySQL/bin

La primera cosa que necesitamos, es crear una base de datos, desde la línea de comando, teclear:

```
mysqladmin -u root create mydb
```

Esto crea la base de datos llamada "mydb". El comando dice a MySQL que estamos haciendo esto como usuario root.

Ahora adicionaremos datos utilizando un ejemplo clásico, una base de datos de empleados. Vamos a necesitar un archivo core dump que mencionamos anteriormente. Si estas interesado en profundizar en el tema revisa el manual que viene con MySQL o revisa la liga <http://www.turbolift.com/mysql/> .

Copia y pega el siguiente texto en un archivo y guárdalo en el directorio c:\mysql\bin. (nombra al archivo mydb.dump.)

```
CREATE TABLE employees ( id tinyint(4) DEFAULT '0' NOT NULL AUTO_INCREMENT, first
varchar(20), last varchar(20), address varchar(255), position varchar(50), PRIMARY KEY
(id), UNIQUE id (id));INSERT INTO employees VALUES (1,'Bob','Smith','128 Here St,
Cityname','Marketing Manager');

INSERT INTO employees VALUES (2,'John','Roberts','45 There St , Townville','Telephonist');

INSERT INTO employees VALUES (3,'Brad','Johnson','1/34 Nowhere Blvd, Snowston','Doorman');
```

Ahora insertaríamos en la base de datos mydb, Desde la línea de comandos tecleamos:

```
mysql -u root mydb < mydb.dump
```

Podrías tener algunos errores ejecutando esto. Si los tienes revisa y busca alguna línea que haya sido escrita de forma incorrecta.

## 1.6. Colocando juntas todas las piezas

Ahora que tenemos los datos en la base de datos. Vamos a hacer algunas cosas con ellos. Copia y pega las siguientes líneas de texto y salva el archivo en un documento dentro del servidor de Web con la extensión .php

```
<html>
<body>

<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
$result = mysql_query("SELECT * FROM employees",$db);
printf("First Name: %s<br>\n", mysql_result($result,0,"first"));
printf("Last Name: %s<br>\n", mysql_result($result,0,"last"));
printf("Address: %s<br>\n", mysql_result($result,0,"address"));
printf("Position: %s<br>\n", mysql_result($result,0,"position"));
?>

</body>
</html>
```

Explicaremos que sucede aquí. La función **mysql\_connect()** abre una conexión al servidor MySQL en el host especificado (en este caso es el host local) accesandolo con el nombre del usuario (root). Si necesitas especificar un password, puedes adicionarlo también. El resultado de la conexión es almacenada en la variable \$db.

**Mysql\_select\_db()** le dice a PHP que las consultas que estamos haciendo van contra la base de datos mydb. Podríamos crear múltiples conexiones a las bases de datos en diferentes servidores. Pero por ahora, dejaremos esto así.

La siguiente, **mysql\_query()** hace todo el trabajo duro. Usando el identificador de la conexión a la base de datos, este envía una línea de SQL al servidor MySQL para ser procesada. Los resultados que son retornados son almacenados en la variable \$result.

Finalmente, **mysql\_result()** es usado para desplegar los valores de los campos de nuestra consulta. Usando \$result, nos vamos a la primera línea, la cual esta numerada como 0 y desplegamos el valor de los campos especificados.

La sintaxis de print puede ser vista como las utilizadas en C o Perl. En cada una de las líneas de encima, los % indican que la variable en la segunda mitad de la expresión (por ejemplo, mysql\_result(\$result,0,"position")) pueden ser manejados como una cadena e imprimirlos. Para más información sobre print, puedes ver la documentación de PHP.

<http://www.php3.net/manual/function.printf.php3>

Hasta aquí hemos compilado, instalado y configurado exitosamente MySQL y PHP; además hemos ejecutado un script simple para recuperar información.

En la unidad 2, con destreza y maña desplegaremos múltiples registros y al mismo tiempo introduciremos datos provenientes de una página web en la base de datos.

## Unidad 2.

### Escribiendo programas con PHP / MySQL

Vamos a continuar con la escritura de programas con PHP y MySQL. Comencemos por desplegar en una página web la base de datos que creamos en la unidad anterior, pero ahora anexemos nuevos elementos de diseño en HTML.

Primero realicemos una búsqueda en la base de datos utilizando el siguiente código:

```
<html>
<body>

<?php
    $db = mysql_connect("localhost", "root");
    mysql_select_db("mydb", $db);
    $result = mysql_query("SELECT * FROM employees", $db);
    echo "<table border=1>\n";
    echo "<tr><td>Name</td><td>Position</tr>\n";
    while ($myrow = mysql_fetch_row($result)) {
        echo "<tr>";
        echo "<td>$myrow[1] $myrow[2]</td>";
        echo "<td>$myrow[3]</td>";
        echo "</tr>";
    }
    echo "</table>\n";
?>

</body>
</html>
```

Probablemente notarás que se han introducido algunas nuevas características en el código anterior. En principio la instrucción **while()**. La instrucción while construye un ciclo, indica que en tanto existan nuevos renglones de datos disponibles (los cuales son leídos de la base de datos mediante la instrucción **mysql\_fetch\_row()**), se almacena el renglón actual en la variable **\$myrow**. Luego se ejecutan las instrucciones encerradas entre las llaves. Observa el código cuidadosamente y seguramente el sentido quedara claro.

Un pequeño problema con **mysql\_fetch\_row()** es que retorna un arreglo que soporta solo referencias numéricas a campos individuales. Así que el primer campo es referido como 0, el segundo como 1 y así sucesivamente. En consultas complejas esto puede ser algunas veces un problema.

Ahora examinemos el ciclo con más detalle. Las primeras líneas las puedes reconocer del ejemplo de la primera unidad. En el ciclo while, tomamos una línea del resultado y la asignamos al arreglo \$myrow. Entonces imprimimos el contenido del arreglo en la pantalla con la función printf. Después este ciclo inicia de nuevo y otra línea es asignada a \$myrow. Esto sucede hasta que obtiene todas las líneas disponibles.

Pero si la consulta no retorna datos, no tenemos manera de que el usuario lo conozca.

#### 2.1. Mantener informado al usuario

Observe el siguiente script:

```
<html>
<body>

<?php
    $db = mysql_connect("localhost", "root");
    mysql_select_db("mydb", $db);
    $result = mysql_query("SELECT * FROM employees", $db);
    if ($myrow = mysql_fetch_array($result)) {
        echo "<table border=1>\n";
        echo "<tr><td>Name</td><td>Position</td></tr>\n";
        do {
            printf("<tr><td>%s %s</td><td>%s</tr>\n", $myrow["first"], $myrow["last"],
                $myrow["address"]);
        } while ($myrow = mysql_fetch_array($result));
        echo "</table>\n";
    } else {
        echo "Sorry, no records were found!";
    }
}
?>

</body>
</html>
```

Hay un número de nuevos desarrollos introducidos aquí pero no son tan simples. Primero, existe la función `mysql_fetch_array()`. Es exactamente lo mismo que `mysql_fetch_row()` con una pequeña excepción: Usando esta función, podemos referirnos a los campos por sus nombre (como `$myrow["first"]`) en lugar de sus números. Esto podría ahorrarnos algunos dolores de cabeza. También introducimos un ciclo `do/while` y una sentencia `if-else`.

Si la sentencia `if-else` dice que si podemos asignar una celda a `$myrow`, entonces que continúe; en otro caso salta a la sección del `else` y hacer lo que hay ahí.

El ciclo `do/while` es una variación del `while()` que utilizamos antes. Necesitamos hacer el `do/while` aquí por una buena razón: Con la sentencia inicial `if`, nosotros asignamos el primer renglón retornado de la consulta a la variable `$myrow`. Si en este punto ejecutamos una sentencia regular `while` (como `while($myrow = mysql_fetch_row($result))`), Tenemos que sacar forzosamente el primer registro fuera de la variable y reemplazándola con el segundo registro. Pero el ciclo `do/while` nos deja probar la condición después de que el código ha sido corrido ya una vez. Así que no hay oportunidad de que nos saltemos algún renglón.

Finalmente, si no hay registros retornados en la consulta, las sentencias contenidas en la porción, `else{}` serán ejecutadas. Para ver esta porción en acción, cambia la sentencia SQL a : `SELECT * FROM employees WHERE id=6` o alguna otra cosa que no retorne registros.

Ahora vamos a extender este ciclo y el código `if-else` para hacer una página más elegante.

## 2.2. Despliegado de información

Vamos a tomar el poder de un ciclo que aprendimos y usamos antes para crear un ejemplo mas practico. Pero antes de que procedamos aquí, usted debe saber trabajar con las formas HTML, el QUERY\_STRING y los métodos GET y POST.

Vamos a trabajar con la información del QUERY\_STRING. Como usted debe saber, hay tres maneras de conseguir la información en la cadena de consulta. Lo primero es utilizar el método del GET en una forma. Lo segundo es el tipo de información dentro del URL en tu Navegador.

Escribe una etiqueta como esta en un archivo HTML:

```
<a href="http://my_machine/mypage.php3?id=1">
```

Vamos a utilizar esta técnica ahora.

Primero consultaremos nuestra base de datos otra vez y enumeraremos los nombres de los empleados. Observe el script siguiente. Mucho de esto se vera muy familiar ahora.

```
<html>
<body>

<?php
    $db = mysql_connect("localhost", "root");
    mysql_select_db("mydb", $db);
    $result = mysql_query("SELECT * FROM employees", $db);
    if ($myrow = mysql_fetch_array($result)) {
        do {
            printf("<a href=\"%s?id=%s\">%s %s</a><br>\n", $PHP_SELF, $myrow["id"],
                $myrow["first"], $myrow["last"]);
        } while ($myrow = mysql_fetch_array($result));
    } else {
        echo "Sorry, no records were found!";
    }
?>

</body>
</html>
```

Todas las cosas son lo mismo excepto la función printf, así que observaremos esto con algún detalle.

Primero observa que cada marca, va precedida por un backslash. El backslash dice a PHP que no debe desplegar el carácter que le sigue, más bien lo trata como parte del código. También nota el uso de la variable \$PHP\_SELF. Esta variable, almacena el nombre y la locación del script, se pasa con cada paginación del PHP. Usando \$PHP\_SELF, podemos asegurar que esto sucederá al igual si el archivo es movido a otro subdirectorío o al igual a otra maquina.

Como se ha mencionado estas ligas podrían recargar la página. La segunda vez, sin embargo, información extra es adicionada a la cadena de consulta.

PHP cuando ve una cadena del tipo "nombre = valor" en el URL de la pagina, crea automáticamente una variable con el nombre y valor que indica la cadena. Este desarrollo nos permite probar si es la primera o segunda vez que lo envía la pagina. Todo lo que tenemos que hacer es pregunta al PHP si la variable, en este caso, \$id existe.

Una vez que conocemos la respuesta a la pregunta, podemos desplegar un grupo diferente de información la segunda vez. He aquí el ejemplo de ello:

```
<html>

<body>

<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
// display individual record
if ($id) {
    $result = mysql_query("SELECT * FROM employees WHERE id=$id",$db);
    $myrow = mysql_fetch_array($result);
    printf("First name: %s\n<br>", $myrow["first"]);
    printf("Last name: %s\n<br>", $myrow["last"]);
    printf("Address: %s\n<br>", $myrow["address"]);
    printf("Position: %s\n<br>", $myrow["position"]);
} else {
    // show employee list
    $result = mysql_query("SELECT * FROM employees",$db);
    if ($myrow = mysql_fetch_array($result)) {
        // display list if there are records to display
        do {
            printf("<a href=\"%s?id=%s\">%s %s</a><br>\n", $PHP_SELF, $myrow["id"],
                $myrow["first"], $myrow["last"]);
        } while ($myrow = mysql_fetch_array($result));
    } else {
        // no records to display
        echo "Sorry, no records were found!";
    }
}
?>

</body>
</html>
```

Este código se esta complicando, así que es buena idea comenzar a hacer comentarios para seguirle la pista a lo que esta sucediendo. Puede utilizar // para hacer un comentario sencillo de una sola línea o utilizar /\* y \*/ para iniciar y finalizar respectivamente un bloque largo de comentario.

Aquí tenemos por fin, un verdadero primer y útil script de PHP/MySQL. Ahora vamos a observar como vamos conectar formas HTML para enviar información a la base de datos.

### 2. 3. Inserción de datos en la base desde una pagina web

Hemos obtenido datos de la base de datos sin mucha dificultad. Pero enviar datos es otra historia.

Primero vamos a crear una página con una forma HTML simple.

```
<html>
```

```

<body>

<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enter information">
</form>

</body>
</html>

```

Note que usamos `$PHP_SELF` otra vez. Como comentamos en la unidad numero 1, se puede utilizar el PHP en cualquier parte dentro de un código HTML. Notara también que cada elemento de la forma concuerda con el nombre del campo en la base de datos. Esto no es obligatorio, pero es una buena idea.

También nota que he dado al botón Submit un atributo nombre. Esto permitirá mas adelante buscar la existencia de una variable `$submit`. De esta manera cuando la página es llamada por segunda vez podremos conocer que ya se ha utilizado la forma HTML.

Tenemos que mencionar que no es necesario tener una página que se recargue así misma. Pueden crearse dos, tres o las páginas que se requieran. Pero el construir una pagina auto referenciada ayuda a mantener todas las cosas compactas.

Vamos a añadir algún código para revisar la entrada a la forma. Leamos el valor del las variables de la forma y coloquémoslas en la pantalla, utilizando `$HTTP_POST_VARS`.

```

<html>
<body>

<?php
    if ($submit) {
        // process form
        while (list($name, $value) = each($HTTP_POST_VARS)) {
            echo "$name = $value<br>\n";
        }
    } else{
        // display form
    }
?>

<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enter information">
</form>

<?php
    }
?>

</body>
</html>

```

Ahora que se observa bien, vamos a tomar la información de la forma y vamos a ponerla en la base de datos.

```

<html>
<body>

```

```

<?php
    if ($submit) {
        // process form
        $db = mysql_connect("localhost", "root");
        mysql_select_db("mydb", $db);
        $sql = "INSERT INTO employees (first,last,address,position)
                VALUES ('$first','$last','$address','$position)";
        $result = mysql_query($sql);
        echo "Thank you! Information entered.\n";
    } else{
        // display form
    }
?>

<form method="post" action="<?php echo $PHP_SELF?>">
First name:<input type="Text" name="first"><br>
Last name:<input type="Text" name="last"><br>
Address:<input type="Text" name="address"><br>
Position:<input type="Text" name="position"><br>
<input type="Submit" name="submit" value="Enter information">
</form>

<?php
    } // end if
?>

</body>
</html>

```

Hemos introducido datos en la base de datos. Pero aun estamos lejos de la perfección. ¿Que pasa si alguien deja un campo en blanco o introduce texto cuando debió de introducir un número? ¿Que pasa si hay un error donde sea?

Vamos ahora a solucionar eso.

## 2.4. Actualizar datos vía una pagina web.

A través de este texto, hemos cargado las declaraciones de SQL en la variable (\$sql) antes de lanzar la consulta a la base de datos con mysql\_query(). Esto es útil para revisar si alguna parte de la sentencia SQL causa un error, ya que siempre podremos desplegar el SQL a la pantalla y examinar los errores.

Ya conocemos como obtener datos de la base de datos. Ahora vamos a intentar modificar los registros que están en la base de datos. Editar datos combina dos elementos que ya hemos visto: desplegar datos en la pantalla y enviar datos a la base de datos vía una forma de entrada. Sin embargo editar es hacer algo diferente a que mostremos los datos apropiados en la forma.

Primeramente, recuperemos el código de la unidad 1 para desplegar los nombres de los empleados en nuestra página. Pero esta vez completos y llenemos nuestra forma con la información de los empleados:

```

<html>
<body>

<?php
    $db = mysql_connect("localhost", "root");
    mysql_select_db("mydb", $db);
    if ($id) {
        // query the DB
        $sql = "SELECT * FROM employees WHERE id=$id";
        $result = mysql_query($sql);
    }

```

```

    $myrow = mysql_fetch_array($result);
?>

    <form method="post" action="<?php echo $PHP_SELF?>">
    <input type=hidden name="id" value="<?php echo $myrow["id"] ?>">
    First name:<input type="Text" name="first" value="<?php echo $myrow["first"] ?>"><br>
    Last name:<input type="Text" name="last" value="<?php echo $myrow["last"] ?>"><br>
    Address:<input type="Text" name="address" value="<?php echo $myrow["address"] ?>"><br>
    Position:
    <input type="Text" name="position" value="<?php echo $myrow["position"] ?>"><br>
    <input type="Submit" name="submit" value="Enter information">
    </form>
<?php
    } else {
        // display list of employees
        $result = mysql_query("SELECT * FROM employees",$db);
        while ($myrow = mysql_fetch_array($result)) {
            printf("<a href='\"%s?id=%s\"'>%s %s</a><br>\n", $PHP_SELF, $myrow["id"],
                $myrow["first"], $myrow["last"]);
        }
    }
?>

</body>
</html>

```

Solo desplegamos la información del campo en el atributo value de cada elemento, lo cual fue mas o menos fácil. Construyamos un poco más. Adicionaremos la habilidad de enviar el código editado de vuelta a la base de datos. De nuevo vamos a utilizar el botón submit para probar si necesitamos procesar la entrada de la forma. Note que las sentencias SQL que utilizamos son un poco diferentes.

```

<html>
<body>

<?php
    $db = mysql_connect("localhost", "root");
    mysql_select_db("mydb",$db);
    if ($id) {
        if ($submit) {
            $sql = "UPDATE employees
                SET first='$first',last='$last',address='$address',position='$position'
                WHERE id=$id";
            $result = mysql_query($sql);
            echo "Thank you! Information updated.\n";
        } else {
            // query the DB
            $sql = "SELECT * FROM employees WHERE id=$id";
            $result = mysql_query($sql);
            $myrow = mysql_fetch_array($result);
?>

            <form method="post" action="<?php echo $PHP_SELF?>">
            <input type=hidden name="id" value="<?php echo $myrow["id"] ?>">
            First name:
            <input type="Text" name="first" value="<?php echo $myrow["first"] ?>"><br>
            Last name:
            <input type="Text" name="last" value="<?php echo $myrow["last"] ?>"><br>
            Address:
            <input type="Text" name="address" value="<?php echo $myrow["address"] ?>"><br>
            Position:
            <input type="Text" name="position" value="<?php echo $myrow["position"] ?>"><br>
            <input type="Submit" name="submit" value="Enter information">
            </form>

<?php
    }

```

```

    } else {
        // display list of employees
        $result = mysql_query("SELECT * FROM employees",$db);
        while ($myrow = mysql_fetch_array($result)) {
            printf("<a href=\"%s?id=%s\">%s %s</a><br>\n", $PHP_SELF, $myrow["id"],
                $myrow["first"], $myrow["last"]);
        }
    }
}
?>

</body>
</html>

```

Y esto es todo. Hemos manejado diferentes combinaciones de características en varios scripts. Obsérvese como hemos utilizado un if() dentro de otro if() para revisar condiciones múltiples.

Es hora de poner todo junto en un script.

## 2.5. Colocar todo junto

Finalizamos esta lección poniendo todo en una página sencilla donde podemos aumentar, editar y remover entradas de la base de datos. Es una extensión de lo que hemos visto hasta ahora y haremos un buen repaso. Echemos un vistazo.

```

<html>
<body>

<?php
$db = mysql_connect("localhost", "root");
mysql_select_db("mydb",$db);
if ($submit) {
    // here if no ID then adding else we're editing
    if ($id) {
        $sql = "UPDATE employees
            SET first='$first',last='$last',address='$address',position='$position'
            WHERE id=$id";
    } else {
        $sql = "INSERT INTO employees (first,last,address,position)
            VALUES ('$first','$last','$address','$position)";
    }
    // run SQL against the DB
    $result = mysql_query($sql);
    echo "Record updated/edited!<p>";
} elseif ($delete) {
    // delete a record
    $sql = "DELETE FROM employees WHERE id=$id";
    $result = mysql_query($sql);
    echo "$sql Record deleted!<p>";
} else {
    // this part happens if we don't press submit
    if (!$id) {
        // print the list if there is not editing
        $result = mysql_query("SELECT * FROM employees",$db);
        while ($myrow = mysql_fetch_array($result)) {
            printf("<a href=\"%s?id=%s\">%s %s</a> \n", $PHP_SELF, $myrow["id"],
                $myrow["first"], $myrow["last"]);
            printf("<a href=\"%s?id=%s&delete=yes\">(DELETE)</a><br>", $PHP_SELF,
                $myrow["id"]);
        }
    }
}
?>

<p>
<a href="<?php echo $PHP_SELF?>">ADD A RECORD</a>

```

```

        <P>
        <form method="post" action="<?php echo $PHP_SELF?>">
<?php
    if ($id) {
        // editing so select a record
        $sql = "SELECT * FROM employees WHERE id=$id";
        $result = mysql_query($sql);
        $myrow = mysql_fetch_array($result);
        $id = $myrow["id"];
        $first = $myrow["first"];
        $last = $myrow["last"];
        $address = $myrow["address"];
        $position = $myrow["position"];
        // print the id for editing
    ?>

        <input type=hidden name="id" value="<?php echo $id ?>">

<?php
    }
?>

First name:<input type="Text" name="first" value="<?php echo $first ?>"><br>
Last name:<input type="Text" name="last" value="<?php echo $last ?>"><br>
Address:<input type="Text" name="address" value="<?php echo $address ?>"><br>
Position:<input type="Text" name="position" value="<?php echo $position ?>"><br>
<input type="Submit" name="submit" value="Enter information">
</form>

<?php
}
?>

</body>
</html>

```

Se ve complejo, pero realmente no lo es. El script esta dividido en tres partes. La primera condición if() checa para ver que el botón de Submit ha sido presionado y si es así, checa que la variable \$id exista. Si no entonces adiciona un registro. En otro caso , edita un registro.

Después revisamos si la variable \$delete existe. Si existe borramos el registro. Nota que con la primera sentencia if() revisamos para una variable que llega vía un POST y en este la variable que podría ser parte de un GET.

Finalmente, tomamos la acción default que despliega la lista de empleados y la forma. De nuevo revisamos la existencia de la variable \$id. Si existe, consultamos a la base de datos para desplegar los registros relevantes. En otro caso, desplegamos una forma en blanco.

Hemos puesto todo lo aprendido y lo pusimos en un script. Usamos ciclos while() y sentencias if() y hemos corrido la gama de sentencias básicas de SQL como SELECT, INSERT, UPDATE y DELETE. Y hemos observado como podemos pasar información de una página a otra usando URLs y formas de entrada.

## Unidad 3.

### Un lugar para todo

Bienvenido a la tercera y parte final de nuestro texto. Si tu has ido a través de la lección 1 y la lección 2 ya conoces lo esencial para instalar y escribir scripts útiles con MySQL y PHP. Vamos a observar algunas funciones de PHP útiles que podrían hacer tu vida más fácil.

#### 3.1. Archivos incluidos.

Primero demos un vistazo en los archivos include.

Un include permite que el contenido de un archivo externo sea referenciado e importados en otro archivo. Para hacer esto en PHP hay dos funciones de las cuales necesitamos hablar: include() y require(). La diferencia entre estas dos funciones es sutil pero importante, así que vamos a verlas de cerca.

La función require() trabaja en así: los archivos son incluidos como parte del documento original tan pronto como el archivo pasa por PHP, sin importa su ubicación. Si se coloca una función require() dentro de un ciclo condicional, el archivo externo podría ser incluido aun si alguna parte del ciclo condicional es falsa.

La función include() importa el archivo referenciado cada vez que es encontrado, si PHP no lo encuentra el incluye es ignorado. Lo que significa que puedes usar include dentro de los ciclos y sentencias condicionales y la ejecución será la misma que la de cualquier otra instrucción.

Finalmente si estas usando require() y el archivo que estas incluyendo no existe el script podría pararse y producir un error. Si usas include(), tu script podría generar un mensaje de advertencia pero seguir ejecutándose. Puedes probar esto tú mismo intentándolo con el siguiente script. Corre el script y luego reemplaza el include() por el require() y compara los resultados.

```
<html>
<body>

<?php
    include("emptyfile.inc");
    echo "Hello World";
?>

</body>
</html>
```

Me gusta usar el sufijo .inc con mis archivos include, así que puedo separarlos de un script normal de PHP.

¿Que uso se recomienda para los archivos include? Simple ! puedes colocar la información común de todas las páginas dentro de estos. Cosas como cabeceras HTML, pies de página, código de conexión a la base de datos y funciones definidas son buenos candidatos para ponerlos ahí. Pega el siguiente texto en un archivo llamado header.inc.

```

<?php
    $db = mysql_connect("localhost", "root");
    mysql_select_db("mydb", $db);
?>

<html>
<head>
<title>
<?php
    echo $title
?>
</title>
</head>
<body>
<center><h2><?php echo $title ?></h2></center>

```

Ahora crea otro archivo llamado footer.txt que contenga algún texto apropiado para cerrar el texto y las banderas HTML.

Ahora vamos a crear el tercer archivo que contenga el script actual de PHP. Intenta el siguiente código, asegurándote que tu servidor MySQL esta corriendo.

```

<?php
    $title = "Hello World";
    include("header.inc");
    $result = mysql_query("SELECT * FROM employees", $db);
    echo "<table border=1>\n";
    echo "<tr><td>Name</td><td>Position</tr>\n";
    while ($myrow = mysql_fetch_row($result)) {
        printf("<tr><td>%s %s</td><td>%s</tr>\n", $myrow[1], $myrow[2], $myrow[3]);
    }
    echo "</table>\n";
    include("footer.inc");
?>

```

¿Que sucede ? Los archivos include son colocados dentro del archivo principal y entonces todo es ejecutado por el PHP. Observa ahora que la variable \$title fue definida antes que sea referenciado header.inc.

Su valor esta disponible en el código de header.inc; desde, que el titulo de la pagina es cambiado. Ahora puedes utilizar header.inc a través de todas tus paginas de PHP y todo lo que tendrías que hacer es cambiar el valor de \$title de pagina a pagina.

Usando una combinación de includes, HTML, sentencias condicionales y ciclos puedes crear variaciones complejas de página a página con un mínimo absoluto de código.

Los includes, son especialmente útiles cuando son utilizados con funciones, como hemos visto en el camino.

### 3.2. Validación simple

Imagina por un momento que hemos estropeado nuestra base de datos y necesitamos información de los usuarios que pudieron insertarse en la base de datos. Adelantándonos, imaginemos que tenemos un campo en nuestra base de datos esperando para alguna entrada numérica, como un precio por así decirlo. Finalmente imagina que tu aplicación falla a causa de algún tipo inteligente que puso texto en este

campo. MySQL no quiere ver texto en la posición donde se espera recibir un valor numérico.

¿Que hacer ?

Es tiempo de validar.

La validación simplemente significa que examinaremos una pieza de datos, usualmente de una forma de HTML para revisarla y asegurarnos que es un modelo apropiado. Esto puede asegurarnos que no es un elemento en blanco para validar que un elemento encuentra cierto criterio (por ejemplo, que un valor numérico es estipulado o que una dirección de mail contiene una @).

La validación puede ser hecha del lado del servidor o del lado del cliente. El PHP es usado para la validación del lado del servidor, mientras que JavaScript u otro lenguaje de scripts se puede emplear para la validación del lado del cliente (en el navegador). Este artículo es acerca de PHP, así que nos vamos a concentrar en las cosas del servidor.

Ignoraremos nuestra base de datos y nos concentraremos en la validación de PHP. Si quieres, puedes adicionar campos a la base de datos de los empleados utilizando las sentencias ALT de MySQL esto si quieres que los datos que validemos sean insertados en la base.

Existen muchas funciones de PHP útiles que podemos usar para validar nuestros datos, pueden ser simples o altamente complejos. Una función simple que podríamos usar podría ser strlen(), la cual nos dice la longitud de la variable.

Una función mas compleja podría ser ereg(), la cual maneja expresiones regulares para consultas complejas. No quiero meterme en complejidades de registros aquí, pero existen libros enteros escritos sobre la materia.

Veamos un ejemplo simple, para revisar si una variable existe o no existe.

```
<html>
<body>

<?php
    if ($submit) {
        if (!$first || !$last) {
            $error = "Sorry! You didn't fill in all the fields!";
        } else {
            // process form
            echo "Thank You!";
        }
    }
    if (!$submit || $error) {
        echo $error;
    }
?>

<P>
<form method="post" action="<?php echo $PHP_SELF ?>">
FIELD 1: <input type="text" name="first" value="<?php echo $first ?>">
<br>
FIELD 2: <input type="text" name="last" value="<?php echo $last ?>">
<br>
<input type="Submit" name="submit" value="Enter Information">
</form>
```

```

<?php
}
?>

</body>
</html>

```

Las claves de este script son las sentencias condicionales anidadas. La primera checa para ver si el botón de submit ha sido presionado. Si es así, se va a revisar si existen las variables \$first y \$last. El símbolo || significa "or" y el símbolo ! significa "not".

Podríamos también escribir la sentencia diciendo, "Si no existe \$first o si \$last no existe, entonces pon \$error a lo siguiente".

Después, extenderemos pocas cosas para revisar si es que una cadena tiene cierta longitud.

Esto podría ser ideal para passwords, desde cuando no desees algún holgazán introduzca un password con una o dos letras. Puedes decirle que sea de seis o más caracteres.

La función para esto tu ya la conoces, strlen(). Simplemente retorna un número igual al número de caracteres en la variable que esta siendo revisada. Aquí, modifique el script de abajo para revisar la longitud de \$first y de \$last.

```

<html>
<body>

<?php
    if ($submit) {
        if (strlen($first) < 6 || strlen($last) < 6) {
            $error = "Sorry! You didn't fill in all the fields!";
        } else {
            // process form
            echo "Thank You!";
        }
    }
    if (!$submit || $error) {
        echo $error;
    }
?>

    <P>
    <form method="post" action="<?php echo $PHP_SELF ?>">
    FIELD 1: <input type="text" name="first" value="<?php echo $first ?>">
    <br>
    FIELD 2: <input type="text" name="last" value="<?php echo $last ?>">
    <br>
    <input type="Submit" name="submit" value="Enter Information">
    </form>

<?php
}
?>

</body>
</html>

```

Corre este script e intenta introducir seis caracteres o menos y observa que sucede. Es simple pero efectivo.

### 3.3. Funciones

Hemos ya usado funciones cientos de veces. Cada vez que consultamos la base de datos o revisamos la longitud de una cadena, estamos usando funciones. Estas funciones son construidas dentro de PHP. Si lo deseas puedes extender el PHP con tus propias funciones personalizadas.

Una función es simplemente un bloque de código al que le pasamos uno o más valores. La función entonces procesa la información y retorna el valor. La función puede ser tan simple o tan compleja como queramos.

Vamos a crear una función. Iniciaremos de manera simple. Necesitamos dar a la función un nombre y decirle que variables esperar. También necesitamos definir la función antes de llamarla.

```
<html>
<body>

<?php
    function addnum($first, $second) {
        $newnum = $first + $second;
        return $newnum;
    }
    echo addnum(4,5);
?>

</body>
</html>
```

Primeramente crearemos nuestra función. Nótese como definimos dos nuevas variables, llamadas \$first y \$second. Cuando llamamos a la función, cada variable es asignada a un valor basado en el orden en el cual aparecen en la lista - 4 es a \$first, 5 es a \$second. Entonces simplemente adicionamos los dos números juntos y retornamos el resultado. "Return" aquí simplemente significa enviar el resultado de regreso. Al final del script imprimimos el numero 9.

Crearemos algunas cosas que serán más útiles para las aplicaciones de base de datos. Como hacer algunos códigos que manejen errores:

```
<html>
<body>

<?php
    function do_error($error) {
        echo "Hmm, looks like there was a problem here...<br>";
        echo "The reported error was $error.\n<br>";
        echo "Best you get hold of the site admin and let her know.";
        die;
    }
    if (!$db = @mysql_connect("localhost","user", "password")) {
        $db_error = "Could not connect to MySQL Server";
        do_error($db_error);
    }
?>

</body>
</html>
```

Antes que ejecutes esto apaga tu servidor de base de datos MySQL. Se desplegará un agradable y útil mensaje de error. Los lectores observadores podrían notar que el símbolo de @ en frente de mysql\_connect(). Esto suprime los mensajes de error, así que si quieres obtener la información solo para la función.

Puedes crear una función para consultar una base de datos y desplegar el grupo de resultados sobre muchas páginas.

```
<html>
<body>

<?php
    function db_query($sql) {
        global $db;
        $result = mysql_query($sql,$db);
        return $result;
    }
    $sql = "SELECT * FROM mytable";
    $result = db_query($sql);
?>

</body>
</html>
```

Otro ejemplo, cuando nos conectamos a la base de datos, siempre nos conectamos al mismo servidor y deseamos utilizar siempre el mismo username y password. Pero en ocasiones quizá necesitemos conectarnos a diferentes bases de datos. Vamos a echar un vistazo.

```
<html>
<body>

<?php
    function db_connect($host = "localhost", $user="username",$pass="graeme") {
        $db = mysql_connect($host, $username, $password);
        return $db;
    }
    $old_db = db_connect();
    $new_host = "site.com";
    $new_db = db_connect($new_host);
?>

</body>
</html>
```

Las variables usadas dentro de la función fueron definidas cuando la función fue creada. La primera vez que la función es llamada, los valores iniciales son utilizados. La segunda vez, nos conectamos a un nuevo host, pero con el mismo username y password.

Piensa acerca de donde podemos utilizar estas funciones en tu código. Podrías usarlas para revisar datos, en tareas rutinarias, y donde sea. Yo las utilizo cuando proceso texto para desplegarlo en una página de Web. Puedo revisar, la escritura y modificar el texto para aumentar nuevas líneas y caracteres de escape HTML en un solo golpe.

### 3.4. Conclusión

Sobre base de datos existe mucho material. Si deseas desarrollar tus habilidades en esta área siempre habrá mucho por aprender. Encuentra un buen libro acerca del diseño de base de datos y aprende a poner una base de datos sólida - en alguna plataforma. Esta es una invaluable destreza y podría ahorrarte mucho tiempo y dolores de cabeza a la larga. También aprende acerca de MySQL. Es una base de datos compleja pero interesante con mucha documentación útil disponible en Internet. Aprende sobre las estructuras de las tablas, tipos de datos, y SQL. SQL es el lenguaje de consulta estándar para bases de datos.

Finalmente continúa con PHP. El sitio de PHP tiene todo lo que tú necesitas, desde un manual comprensible hasta una lista de mail y grupos de discusión. Una excelente manera de aprender PHP es estudiando los ejemplos usados en el manual y también observando ejemplos disponibles en diversos sitios en la web. Muchos scripts disponibles en el sitio oficial de PHP consisten en funciones o clases que puedes usar gratis en tus propios scripts, sin que tengas que reinventar la rueda. Adicionalmente, las listas de correo es una excelente forma de anunciarte y obtener ayuda. Los desarrolladores mismos leen la lista y hay bastante gente con conocimiento que te pueda ayudar a lo largo del camino.

Buena suerte y buena codificación.

#### Crédito:

Elaboración: [Graeme Merrall \(graeme@kc.co.nz\)](mailto:graeme@kc.co.nz)

Traducción: [Eliud Herrera Rojas \(eliud@mixteco.utm.mx\)](mailto:eliud@mixteco.utm.mx)

Guía de Instalación: [Javier González \(javiergs@acm.org\)](mailto:javiergs@acm.org)