

FUNDAMENTOS DE PROGRAMACIÓN CON



FAVA - Formación en Ambientes Virtuales de Aprendizaje

SENA - Servicio Nacional de Aprendizaje.

2. Usos prácticos de php



A diferencia de Javascript que se ejecuta del lado el cliente (client-side programming) PHP es un lenguaje que se ejecuta del lado del servidor (server-side programming) y permite la generación de páginas web dinámicas, es decir, páginas que se crean de acuerdo a la lógica del momento en que se llaman.

Al estar del lado del servidor PHP se puede comunicar fácilmente con bases de datos y otros recursos que permiten que las aplicaciones PHP realicen las mismas tareas que pueden ejecutar las aplicaciones nativas o de escritorio.

A continuación algunas tareas importantes que se pueden realizar con PHP.

2.1. Formularios web

En el recurso introductorio al diseño web de ADSI se introdujo al aprendiz del concepto de formularios web, sin embargo, estos formularios no tenían mayor funcionalidad ya que los datos capturados no podían ser enviados a un servidor para su procesamiento. Con PHP se pueden tomar esos datos y procesarlos ya sea a través de un programa convencional o interactuando con recursos del sistema como bases de datos, archivos del filesystem, entre otros.

2.1.1. Paso de variables a un programa PHP

Para el desarrollo de formularios con PHP se requiere primero entender cómo se pasan las variables desde el formulario web hacia el programa PHP que hace el procesamiento.



Existen dos maneras de pasar variables desde una página web a PHP: con el método GET y con el método POST.

a. Método GET usando una URL

Mediante este sistema las variables se pasan a un programa PHP usando la URL. Por ejemplo:

```
<a href="holamundo.php?nombre=pedro&apellido=perez"> Enlace a
Pedro Pérez </a>
```

En el ejemplo se están pasando los parámetros “nombre” y “apellido” al programa “holamundo.php” a través de una URL usando el elemento HTML ancla (<a>). De la expresión se tiene que:

El símbolo “?” indica el inicio de los parámetros.

El símbolo “&” es el separador de parámetros.

El símbolo “=” indica el valor de la variable.

b) Método GET a través de un formulario

Una estructura básica de un formulario HTML procesado con un programa PHP es:

```
<form action="<nombre_programa>.php">
  Enunciado del campo:
  <input type="text" name="campo1">
  <input type="submit" value="Enviar">
</form>
```

Donde el atributo “action” indica el programa al cual se le enviarán las variables contenidas en los campos input. Si se omite el atributo “action” el formulario enviará las variables al mismo programa, es decir, en lugar de hacer un llamado a un programa externo como: “otro_programa.php?parametro1=uno¶metro2=dos” el sistema pasará los parámetros al mismo programa así: “mismo_programa.php?parametro1=uno¶metro2=dos” y recargará la página.

Lo anterior se ilustra con el siguiente ejemplo:

```

1  <!--formulario.html -->
2  <html>
3      <body>
4          <form action="programa1.php">
5              Nombre: <input type="text" name="nombre"><br>
6              Apellido: <input type="text" name="apellido"><br>
7                  input type="submit" value="Enviar">
8          </form>
9      </body>
10 </html>

```

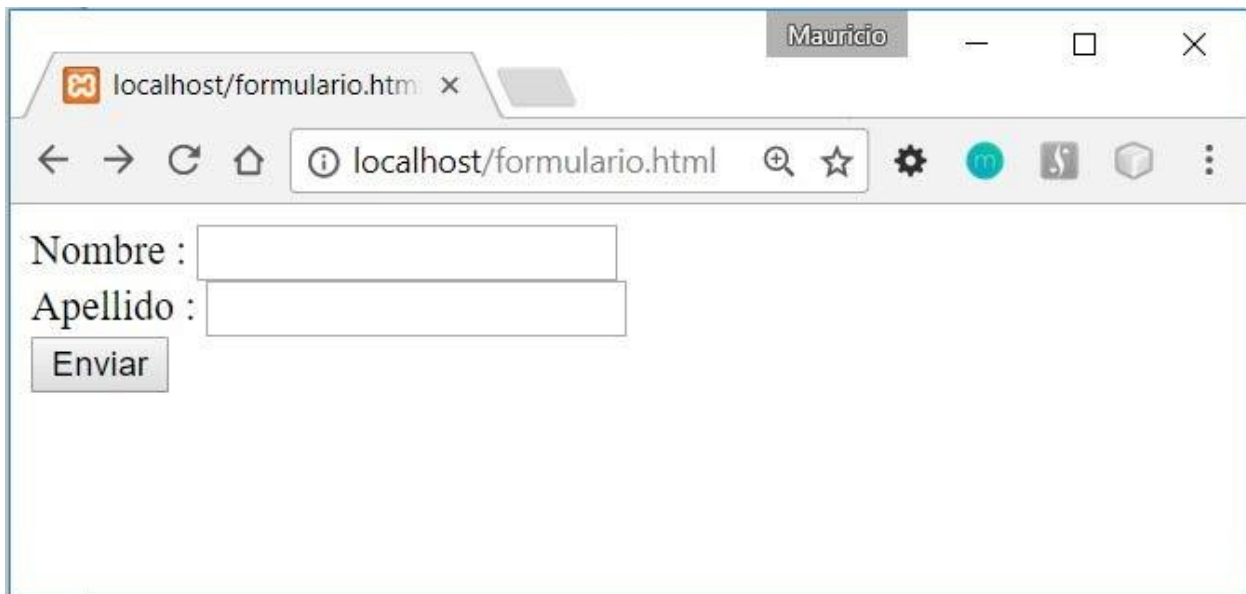
```

1  <!--programa1.php-->
2  <html>
3      <body>
4          <?php>
5              echo "Hola".$_POST["nombre"]." ".$_POST["apellido"]."<br>";
6          <?>
7      </body>
8  </html>

```

Al solicitar al navegador la página: <http://localhost/formulario.html>.

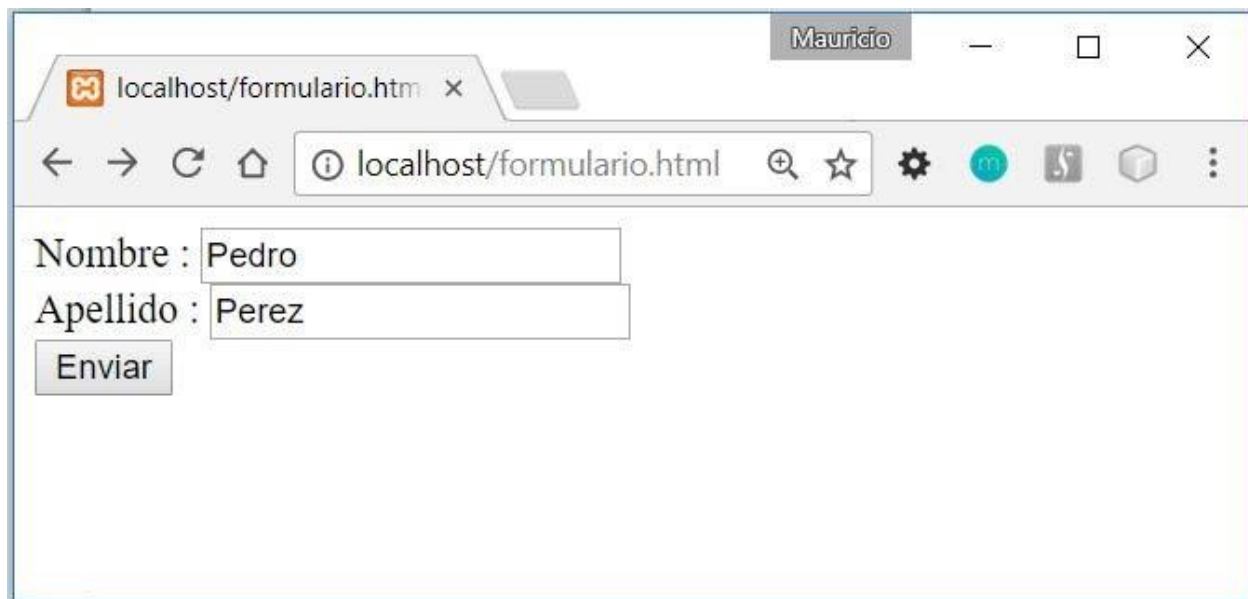
Aparece lo siguiente:



The screenshot shows a web browser window with the title 'Mauricio'. The address bar displays 'localhost/formulario.html'. The page content includes a form with two text input fields labeled 'Nombre :' and 'Apellido :', and a button labeled 'Enviar'.

Figura 15. Página web con un formulario.

Al llenar campos con datos de ejemplo se tiene lo siguiente:



A screenshot of a web browser window. The address bar shows 'localhost/formulario.htm'. The page contains a form with two text input fields. The first field is labeled 'Nombre :' and contains the text 'Pedro'. The second field is labeled 'Apellido :' and contains the text 'Perez'. Below these fields is a button labeled 'Enviar'.

Figura 16. Página web con un formulario diligenciado.

Al oprimir el botón de enviar se obtiene lo siguiente:

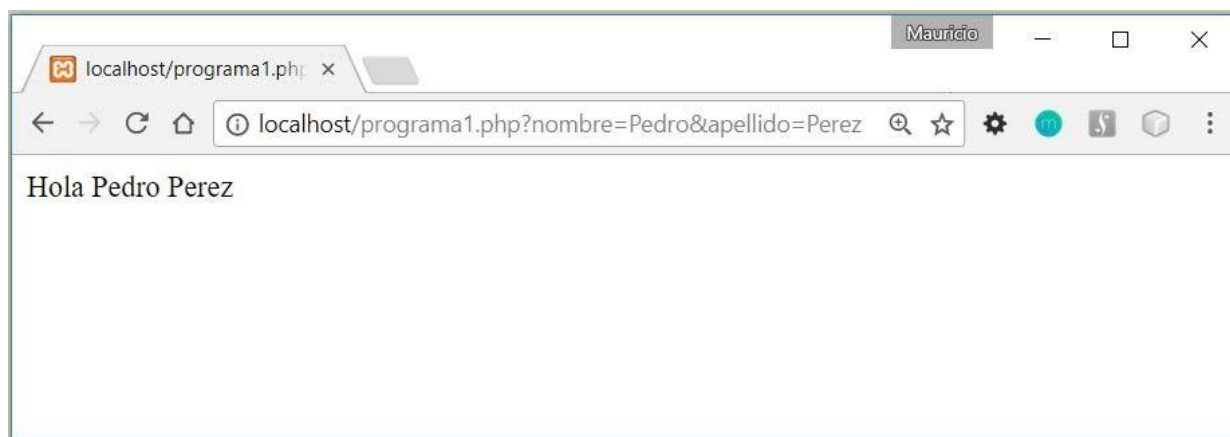


Figura 17. Página resultante de programa1.php usando método GET.

Se puede observar que las variables del formulario aparecen en la URL de la página. También se observa que en la página se dibujó el resultado del código PHP.

c. Formulario con método POST

El método por defecto usado por los formularios para el envío de las variables de un formulario es GET. Lo anterior quiere decir que si no se especifica el método el navegador usará el método GET.

En el método POST las variables no se anexan a la URL sino que se anexan al cuerpo de la solicitud del protocolo HTTP ("HTTP request") y por tanto no son visibles en la barra de navegación.

En el ejemplo anterior si se usa el método POST el programa queda como sigue:

```
1  <!--formulario.html -->
2  <html>
3  <body>
4  <form action="programa1.php"method=post>
5      Nombre: <input type=text name="nombre"><br>
6      Apellido: <input type=text name="apellido"><br>
7      <input type=submit value="Enviar">
8  </form>
9  </body>
10 </html>
```

```
1  <!--programa1.php-->
2  <html>
3  <body>
4  <?php>
5      echo "Hola".$_POST["nombre"]." ".$_POST["apellido"]."<br>";
6  <?>
7  </body>
8  </html>
```

De los anteriores programas se tiene:

1. En la definición de la forma en el archivo html se especificó el atributo "method=post".
2. En el programa PHP la variable que se consulta es \$_POST.

Al abrir el formulario en el navegador y oprimir el botón enviar se obtiene lo siguiente:

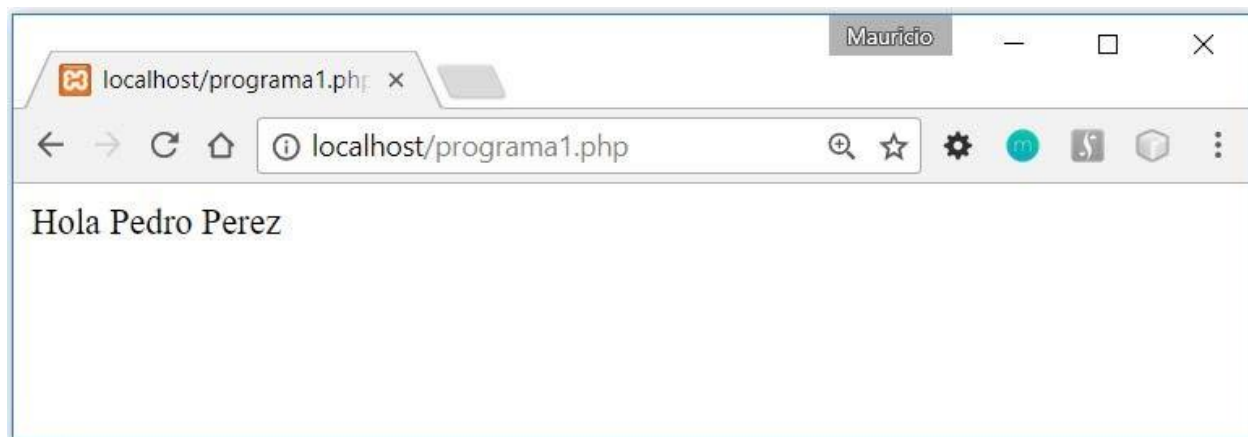


Figura 18. Página resultante de programa1.php usando el método POST.

Se puede observar que en la barra de navegación no aparecen los parámetros. Los parámetros pueden ser enviados por cualquiera de los dos métodos sin embargo se debe tener en cuenta lo siguiente:

- La longitud de una URL está limitada a 3000 caracteres incluyendo los parámetros. Por tanto si se requiere enviar información de una forma que sobrepase ese número se debe usar el método POST.
- Los datos enviados con el método GET son visibles en la barra de navegación. Por lo anterior si se está manejando información sensible o confidencial no es buena opción usar este método.
- Las URL usadas en el método POST no se pueden guardar como un “bookmark”.

2.2. Acceso a bases de datos desde PHP

Una de los beneficios de usar PHP es que ya trae incorporado el código fuente para acceder a la mayoría de las bases de datos.

El flujo que se sigue para el procesamiento de páginas web con código PHP incorporado que acceden a bases de datos es como se muestra en la figura 19.

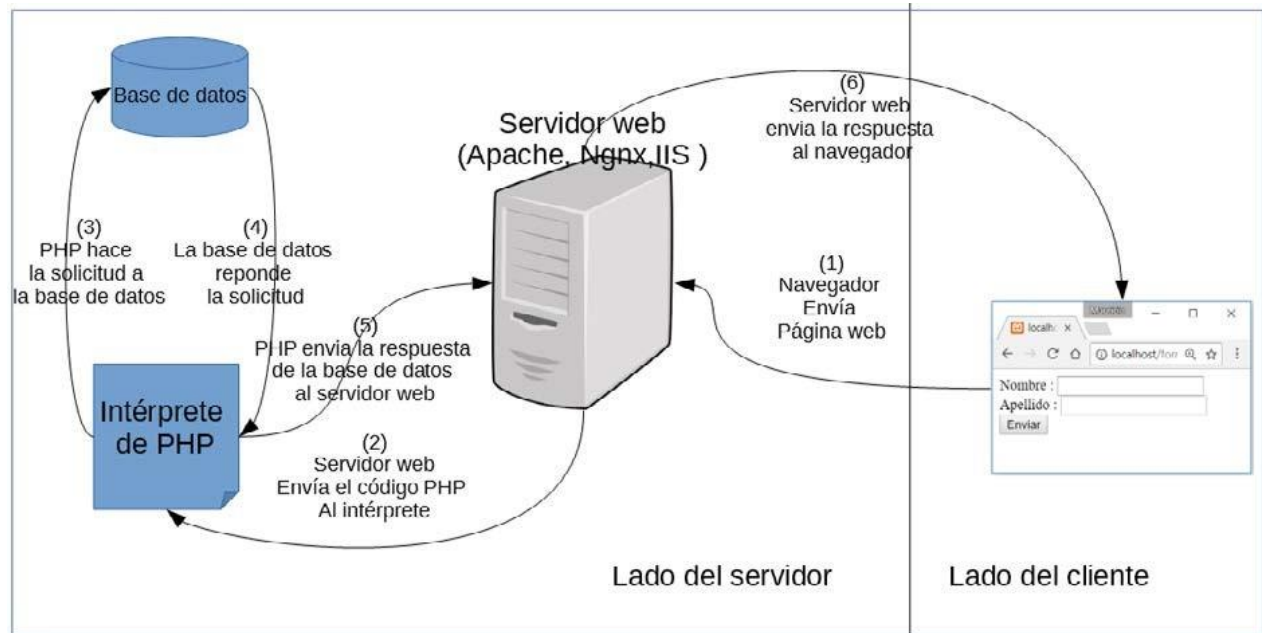


Figura 19. Esquema de operación de PHP con bases de datos.

El proceso para interactuar con bases de datos usando PHP es así:

- Conectarse a la base de datos.
- Seleccionar la base de datos a usar.
- Construir o armar la sentencia SQL.
- Ejecutar la consulta.
- Procesar los datos recibidos.
- Ejecutar los pasos c) al e) hasta que se requiera.
- Desconectarse de la base de datos.

Para desarrollar los ejemplos de este recurso se ha creado una base de datos de prueba con el siguiente script:

2.2.1. mysqli o Mysqli Improved

```
create database adsi ;
create table ciudades ( codigo text, nombre text ) ;
insert into ciudades ( codigo, nombre ) values ( "05001", "MEDELLIN" );
insert into ciudades ( codigo, nombre ) values ( "05002", "ABEJORRAL" );
insert into ciudades ( codigo, nombre ) values ( "05004", "ABRIAQUI" );
insert into ciudades ( codigo, nombre ) values ( "05021", "ALEJANDRIA" );
```

Las librerías para acceder a bases de datos usando PHP son:



Es una librería propia para la base de datos MySQL que, al momento de escribir el recurso, está soportada y en ciclo de vida activo. Fue introducida con la versión 5 de PHP.

Entre sus características están:

- a. Interfaz orientada a objetos.
- b. Tiene un lenguaje procedural.
- c. Permite conexiones persistentes.
- d. Soporta procedimientos almacenados.
- e. Soporte para transacciones.

El siguiente es la estructura para acceder a una base de datos MySQL usando la librería `mysqli`.

En el ejemplo se tiene lo siguiente:

```
1 $conn = new mysqli("<host>", "<usuario>", "<clave>", "<base_de_datos>" ) ;
2     if( $conn->connect_errno ) {
3         echo "Falla al conectarse a Mysql ( ". $conn->connect_errno . " ) " .
5             $conn->connect_error ;
6     } else {
7         echo $conn->host_info. "\n" ;
8     } ;
```

- 1. El parámetro “host” es el nombre o la IP del servidor MySQL.
- 2. El parámetro “usuario” es el usuario de MySQL.
- 3. El parámetro “clave” es la clave del usuario en MySQL.
- 4. El parámetro “base de datos” es el nombre de la base de datos en MySQL que se quiere acceder.

La librería `mysqli` tiene dos formas de operar: con programación procedural o con programación orientada a objetos. En este recurso se usará la orientada a objetos.

Una vez establecida la conexión se pueden realizar las siguientes operaciones:

```

1 <?php
2     $conn = new mysqli("localhost", "desarrollador", "adsi2017", "citas" ) ;
3     if( $conn->connect_errno) {
4         echo "Falla al conectarse a Mysql ( ". $conn->connect_errno . " ) " .
5         $conn->connect_error ;
6     } else {
7         echo $conn->host_info. "\n" ;
8     } ;
9 ?>

```

```

1 <?php
2     $conn = new mysqli("localhost", "desarrollador", "adsi2017", "adsi" ) ;
3     if( $conn->connect_errno) {
4         echo "Falla al conectarse a Mysql ( ". $conn->connect_errno . " ) " .
5         $conn->connect_error ;
6     } else {
7         echo $conn->host_info. "\n" ;
8     } ;
9 ?>

```

Si la conexión es exitosa aparece lo siguiente en una consola PHP:

```

USUARIO@DESKTOP-7971D4H d:\xampp
# php database1.php
localhost via TCP/IP

USUARIO@DESKTOP-7971D4H d:\xampp
#

```

Si la conexión no es exitosa aparece lo siguiente en pantalla:

```

USUARIO@DESKTOP-7971D4H d:\xampp
# php database1.php
Falla al conectarse a Mysql ( 1045) Access denied for user 'desarrollador'@'localhost'
(using password: YES)
USUARIO@DESKTOP-7971D4H d:\xampp
#

```

a. Ejecutar sentencias SQL que no devuelven registros

Para ejecutar una sentencia SQL se usa el método "query". La sintaxis es la siguiente:
En el ejemplo anterior el resultado de la consulta queda almacenado en el objeto

```
if( $mysqli->query("<sentencia sql>") === TRUE ) {
    echo "Se ejecutó la sentencia con éxito";
} else {
    echo "Hubo un error ..." ;
};
```

b. Ejecutar sentencias SQL que devuelven registros

```
if( $resultado = $mysqli->query("<sentencia sql>") ) {
    echo "La consulta devolvió ". $resultado->num_rows . " registros";
} else {
    echo "Hubo un error ..." ;
};
```

“\$resultado .

c. Recorrer los registros devueltos en una consulta con el método fetch_assoc

Para recorrer los registros se pueden usar varios métodos. Uno de ellos es “fetch_assoc” que retorna cada registro como un arreglo asociativo. A continuación un ejemplo: Del código anterior se tiene:

```
01<?php
02 $conn = new mysqli("localhost", "desarrollador", "adsi2017", "adsi" ) ;
03 if( $conn->connect_errno ) {
04     echo "Falla al conectarse a Mysql ( ". $conn->connect_errno . " ) " .
05     $conn->connect_error ;
06     exit() ;
07 } ;
08
09 if($resultado = $conn->query("select codigo, nombre from ciudades ") ){
10
11     while($registro = $resultado->fetch_assoc() ){
12
13         echo $registro["codigo"] . " " . $registro["nombre"] . "\n" ;
14
15     } ;
16 };
17
18 $resultado->free();
19 $conn->close();
20?>
```

- De las líneas 2 al 7 se realiza la conexión a la base de datos.
- La línea 6 indica que si no se pudo conectar a la base de datos termine abruptamente el programa.
- La línea 9 pregunta si el query arrojó resultados. Si está vacío no se entra al ciclo while.
- En las líneas 11 al 15 se ejecuta el ciclo while si el resultado devolvió uno o más registros.
- En la línea 11 se utiliza la variable “\$registro” que se convertirá en un arreglo asociativo una vez se ejecute el método “fetch_assoc”.
- En la línea 13 se extraen los campos del arreglo asociativo y se muestran en el dispositivo de salida.
- En la línea 18 se libera la memoria utilizada por el objeto “\$resultado”.
- En la línea 19 se cierra la conexión.

Al ejecutar el código anterior en la consola de PHP se obtiene lo siguiente:

```
# php database3.php
05001 MEDELLIN
05002 ABEJORRAL
05004 ABRIAQUI
05021 ALEJANDRIA

USUARIO@DESKTOP-7971D4H d:\xampp\htdocs
```

El mismo código se puede ejecutar dentro de una página web, solo se debe incrustar dentro de la sección <body> así:

```
<html>
  <body>
    <programa php>
  </body>
</html>
```

Para este ejemplo también se debe cambiar el salto de línea “\n” por el elemento HTML “
” o break para que tenga el mismo efecto.

Luego se debe copiar el archivo en el directorio “htdocs” del directorio raíz de “xampp”. Al abrir la página con un navegador se obtiene lo siguiente:



Figura 20. Resultado del programa database3.php.

d. Recorrer los registros devueltos en una consulta con el método fetch_object

En este ejemplo se usó el método “fetch_assoc” el cual retorna un arreglo asociativo. PHP también provee el método “fetch_object” el cual retorna los registros como un objeto. Si se hiciera el ejemplo anterior con este método el ciclo while que recorre los registros queda así:

```
09  if($resultado = $conn->query("select codigo, nombre from ciudades ") ){
10
11      while($registro = $resultado->fetch_object() ){
12
13          echo $registro->codigo . " " . $registro->nombre . "\n" ;
14
15      } ;
16  };
```

Se puede observar que la variable “\$registro” ya no es un arreglo asociativo sino un objeto y los campos se convierten en atributos del mismo.

e. Cerrar la consulta

Una vez se hayan procesado los registros y no se requieran más los resultados se debe cerrar el objeto con el método “close” así:

```
$resultado->close();
```

f. Cerrar la conexión

Cuando no se necesite más la conexión ésta se debe cerrar con el método “close” así:

```
$mysqli->close() ;
```

2.2.2. Librería mysql

Esta librería está descontinuada y no se debería usar para proyectos nuevos. Se deben usar las demás librerías mencionadas en este recurso.

2.2.3. Librería PDO

Además de mysqli PHP también suministra la librería PDO o PHP Data Objects que es orientada a objetos y además de MySQL también permite el acceso a otras bases de datos como Postgresql, Oracle, Informix, Firebase, entre otras.

Este recurso hace uso de la librería mysqli y la información acerca de la librería PDO es sólo de interés general.

2.3. Manejo de sesiones

En el desarrollo de software muchas veces el programador requiere manejar datos a lo largo de toda una aplicación. Aunque utilizar una base de datos es una solución posible no siempre es práctico o no siempre se dispone de una base de datos.

En el caso de PHP el código se ejecuta en el servidor el cual puede estar atendiendo muchos usuarios al mismo tiempo. Por otra parte el protocolo HTTP sobre el cual se apoya PHP es “stateless”, es decir, no conoce quién está haciendo las peticiones, sólo las atiende y luego se “olvida”.

Una primera solución a este problema fueron las “cookies” que son unos pequeños archivos que se almacenan en el computador del cliente y almacenan datos propios de ese usuario.

Las “cookies” al ser archivos que se almacenan en el computador cliente y el servidor no tiene control sobre ellas. Por lo anterior pueden ser borradas o alteradas.

Para resolver la anterior problemática PHP suministra una variable superglobal llamada `$_SESSION` el cual es un arreglo asociativo que mantiene mientras se ejecuta una aplicación.

Los pasos para apoyar la programación con base en esta variable superglobal es la siguiente:

a. Iniciar o reiniciar sesión

Para iniciar una sesión se utiliza la instrucción “`session_start()`” en cada programa PHP. Esta instrucción indica a PHP que haga disponible la variable `$_SESSION` en el programa actual.

b. Crear, consultar o modificar elementos del arreglo asociativo \$_SESSION

Para crear un elemento solo se le debe asignar un valor de la misma manera que se hacen con todas las variables en PHP. Por ejemplo: `$_SESSION["usuario"] = "desarrollador"`.

Si el elemento "usuario" de la variable `$_SESSION` ya estaba creada entonces su valor se modificará al último ingresado.

El arreglo `$_SESSION` se puede consultar como cualquier otro arreglo asociativo simplemente accediendo al elemento a través de su nombre. Ejemplo: `echo $_SESSION["usuario"]`.

c. Terminación de la sesión

Una vez se termine la sesión se debe proceder a vaciar la variable `$_SESSION` con la instrucción: `session_unset()` seguido por `session_destroy()`. Lo anterior libera los recursos de memoria que el servidor empleaba para almacenar esos datos.

Glosario

Apache: servidor web de amplio uso para alojar páginas web.

Bookmark: enlace a Internet o URL que se guarda en texto plano para usarlo posteriormente.

Cookie: archivo pequeño que contiene información del cliente.

CSS: acrónimo de Cascade-Style Sheet. Hojas de estilos en cascada.

CUI: acrónimo de Characters User Interface. Interfaz de usuario de caracteres.

GUI: acrónimo de Graphics User Interface. Interfaz gráfica de usuario.

IIS: acrónimo de Internet Information Services. Servidor web desarrollado por Microsoft.

HTML: acrónimo de HyperText Markup Language. Lenguaje de marcación de hipertextos.

HTTP: acrónimo de HyperText Transfer Protocol. Protocolo para el envío y recepción de páginas web.

Javascript: lenguaje de programación interpretado, que se ejecuta del lado del cliente (navegador web).

Librería: conjunto de archivos con programas que proveen servicios a las aplicaciones.

Nginx: servidor web de amplio uso para alojar páginas de web.

PDO: acrónimo de PHP Data Objects. Tecnología de PHP para conectarse a bases de datos.

PHP: acrónimo de Hypertext Preprocesor. Lenguaje de programación para internet.

Unario: operador aritmético o lógico que opera solamente sobre un argumento u operando.

Bibliografía

Cowburn P, (Ed) 2017. *PHP Manual*. Recuperado de <http://php.net/manual/en/>

Nixon, R. 2015. *Learning PHP, MySQL & Javascript Fourth Edition*. O'reilly. Cambridge.

Control del documento

CONSTRUCCIÓN OBJETO DE APRENDIZAJE 	FUNDAMENTOS DE PROGRAMACIÓN EN PHP	
	Centro Industrial de Mantenimiento Integral - CIMI Regional Santander	
	Líder línea de producción:	Santiago Lozada Garcés
	Asesores pedagógicos:	Rosa Elvia Quintero Guasca Claudia Milena Hernández Naranjo
	Líder expertos temáticos:	Rita Rubiela Rincón Badillo
	Experto temático:	Magda Milena García G. (V1) Nelson Mauricio Silva M. (V2)
	Diseño multimedia:	Catalina Martínez Ávila
	Programador:	Francisco José Lizcano Reyes
	Producción de audio:	Víctor Hugo Tabares Carreño

creative commons



BY NC SA

Este material puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de la licencia que el trabajo original.



Registered trademark

Copyright (c) 2017, XAMPP.

EasyPHP 2000 - 2017.

MySQL - licencia dual GNP/Licencia Comercial © ORACLE Corporation. 2017. Todos los derechos reservados.