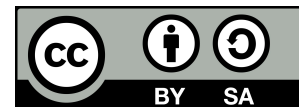


CURSO .PHP

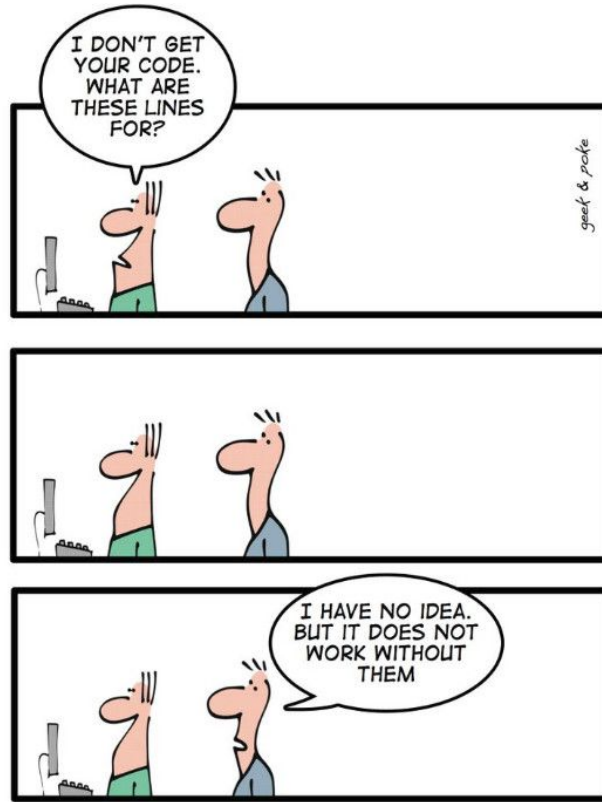
ACCESO A DATOS



Autor: Jon Vadillo
www.jonvadillo.com

Contenidos

- Ejecutar sentencias SQL
- Establecer conexión
- Preparar la sentencia
- Cerrar conexión
- Tratar los resultados






Author: [Geek and Poke](#)

Pre-requisito

- Conocimientos básicos de SQL: CREATE TABLE, SELECT, UPDATE, ...
- ¿Necesitas una ayuda? Añade a tus favoritos el siguiente repositorio:
<https://github.com/jvadillo/guia-rapida-mysql>

🔗 Guía rápida de MySQL

Open Source  Maintained?  yes Ask me  anything

Guía rápida de MySQL donde podrás encontrar la información necesaria y los comando más utilizados.

Conocimientos previos

Esta guía no pretende explicar los conceptos básicos de las bases de datos relacionales, por lo que está enfocada a personas que ya cuenten con un conocimiento básico sobre bases de datos relacionales.

Tabla de contenido

- [Configuración](#)
- [Conexión](#)
- [Gestión de usuarios](#)
- [Bases de datos](#)
- [Tablas](#)
- [Registros](#)
- [Relaciones entre tablas](#)

Ejecutar sentencias SQL

1. Establecer conexión con la base de datos
2. Preparar la sentencia
3. Ejecutar sentencia (asociando parámetros si fuese necesario)
4. Opcional: tratar el resultado de la sentencia ejecutada.

1. Establecer conexión

```
function connect($host, $dbname, $user, $pass){  
    try {  
        # MySQL  
        $dbh= new PDO("mysql:host=$host;dbname=$dbname", $user, $pass);  
        return $dbh;  
    }  
    catch(PDOException $e) {  
        echo $e->getMessage();  
    }  
}
```

2. Preparar la sentencia (sin ejecutarla)

```
$stmt = $dbh->prepare("SELECT nombre, apellidos FROM alumnos" );

// Incluir parámetros con la sintaxis :nombre
$stmt = $dbh->prepare("
    SELECT nombre, apellidos
    FROM alumnos
    WHERE edad > :edad" );

// Otro ejemplo de un INSERT
$stmt= $dbh->prepare("
    INSERT INTO alumnos(nombre, apellidos)
    values (:nombre, :apellidos)" );
```

3. Ejecutar la sentencia

```
// Ejemplo simple sin parámetros
$stmt = $dbh->prepare("SELECT nombre, apellidos FROM alumnos" );
$stmt->execute();

// Ejemplo con parámetros
$data = array( 'nombre' => 'Mikel', 'edad' => 15 );
$stmt = $dbh->prepare("
    SELECT nombre, apellidos
    FROM alumnos
    WHERE nombre = :nombre AND edad = :edad" );
$stmt->execute($data);
```

Placeholders

Cerrar conexión

```
function close(){  
    /**  
     * Una conexión a base de datos con PDO  
     * permanecerá abierta mientras exista  
     * el objeto PDO creado  
     * */  
    $dbh = null;  
}
```

Para cerrar la conexión, debes destruir el objeto asegurándote de eliminar todas las referencias restantes. Esto se logra **asignando null** a la variable que contiene el objeto. Si no lo haces explícitamente, PHP cerrará automáticamente la conexión al finalizar el script.

4. Tratamiento de resultados

- Una vez ejecutada la sentencia `$stmt->execute()` podremos acceder a los resultados obtenidos de la base de datos mediante el objeto **\$stmt**.
- PDO ofrece la posibilidad de recibir los resultados en distintos formatos:
Para indicar el modo se utiliza el método **setFetchMode(String mode)**:
 - **PDO::FETCH_ASSOC**: devuelve un **array asociativo** donde las claves serán los nombres de las columnas.
 - **PDO::FETCH_CLASS**: Asigna los valores de las columnas a las **propiedades de la clase** indicada.
 - **PDO::FETCH_OBJ**: devuelve **objetos anónimos** que tendrán como **propiedades** las columnas obtenidas.

fetch()

- Una vez indicado el cómo queremos los datos, utilizaremos el método `fetch()` para acceder a la información.
- El método `fetch()` obtiene la siguiente fila de un conjunto de resultados, por lo que se deberá iterar por los resultados.

```
$fila = $stmt->fetch(PDO::FETCH_ASSOC);
```

- Si en la base de datos hay un alumno Mikel García, la variable `$fila` tendrá algo así:

```
array(  
    "nombre" => "Mikel",  
    "apellidos" => "García"  
)
```

FETCH_ASSOC

```
$data = array( 'nombre' => 'Mikel', 'edad' => 15 );
$stmt = $dbh->prepare("
    SELECT nombre, apellidos FROM alumnos
    WHERE nombre = :nombre AND edad = :edad");

// Establecemos el modo en el que queremos recibir los datos
$stmt->setFetchMode(PDO::FETCH_ASSOC);

// Ejecutamos la sentencia
$stmt->execute($data);

// Mostramos los resultados obtenidos
while($row = $stmt->fetch()) {
    echo $row['nombre'] . "--";
    echo $row['apellidos'] . "--";
    echo $row['edad ' ] . "<br>";
}
```

FETCH_OBJ

```
$data = array( 'nombre' => 'Mikel', 'edad' => 15 );
$stmt = $dbh->prepare("
    SELECT nombre, apellidos FROM alumnos
    WHERE nombre = :nombre AND edad = :edad");

$stmt->setFetchMode(PDO::FETCH_OBJ);
$stmt->execute($data);

while($row = $stmt->fetch()) {
    echo $row->nombre . "--";
    echo $row->apellidos . "--";
    echo $row->edad . "<br>";
}
```

Hands on!

1. Crea la tabla alumno con los siguientes campos: id (auto-incremental), nombre, apellidos, email y edad. Inserta varios registros manualmente para que contenga algunos datos.
2. Crea una conexión a la base de datos desde PHP.
3. Crea una consulta y muestra los resultados por pantalla.
4. Inserta un nuevo registro desde PHP y vuelve a consultar los datos.

Hands on!

```
CREATE TABLE alumnos (  
    id INT NOT NULL AUTO_INCREMENT,  
    nombre VARCHAR(50) NOT NULL,  
    apellidos VARCHAR(100) NOT NULL,  
    email VARCHAR(100) NOT NULL,  
    edad INT NOT NULL,  
    PRIMARY KEY (id)  
);  
  
INSERT INTO alumnos (nombre, apellidos, email, edad)  
VALUES ('Ane', 'Eguiluz Larrain', 'ane@egibide.org', 25);  
INSERT INTO alumnos (nombre, apellidos, email, edad)  
VALUES ('Jone', 'Zabala Lopez', 'jone@egibide.org', 22);
```

Host: 127.0.0.1 Base de datos: ejemplo2 Tabla: alumnos Datos Consulta*

Básico Opciones Índices (1) Llaves foráneas (0) Comprobar restricciones (0) Particiones Código CREATE Código A

+ Agregar
- Borrar
X Limpiar
▲ Subir
▼ Bajar

Nombre
PRIMARY KEY
id

Tipo / Longitud
PRIMARY

Columnas: + Agregar - Borrar ▲ Subir ▼ Bajar

| # | Nombre | Tipo de datos | Longitud/Con... | Sin signo | Permitir ... | Rellen... | Predeterminado | Comentario |
|---|-----------|---------------|-----------------|--------------------------|--------------------------|--------------------------|-----------------------|------------|
| 1 | id | INT | 10 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | AUTO_INCREM... | |
| 2 | nombre | VARCHAR | 100 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Sin valor predeter... | |
| 3 | apellidos | VARCHAR | 100 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Sin valor predeter... | |
| 4 | email | VARCHAR | 100 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Sin valor predeter... | |
| 5 | edad | INT | 10 | <input type="checkbox"/> | <input type="checkbox"/> | <input type="checkbox"/> | Sin valor predeter... | |

FETCH_CLASS

```
class Alumno {  
  
    public $nombre;  
    public $apellidos;  
    public $edad;  
    public $otraInformacion;  
  
    function __construct($otraInformacion= '') {  
        // El constructor se ejecutará después de asociar los valores  
        // obtenidos de la base de datos al objeto. Por lo tanto, podemos tratar  
        // esos valores dentro del constructor.  
        $this->nombre = strtoupper($this->nombre);  
        $this->otraInformacion = $otraInformacion;  
    }  
}
```

FETCH_CLASS

```
$data = array( 'nombre' => 'Mikel', 'edad' => 15 );
$stmt = $dbh->prepare("SELECT nombre, apellidos FROM alumnos WHERE
nombre = :nombre AND edad = :edad" );
// Establecemos el modo en el que queremos recibir los datos
$stmt->setFetchMode(PDO::FETCH_CLASS, 'Alumno');
// Ejecutamos la sentencia
$stmt->execute($data);

// Mostramos los resultados
while($obj = $stmt->fetch()) {
    echo $obj->nombre;
}
```

Método abreviado query()

- En consultas que no reciban parámetros, podemos utilizar el método abreviado `query()` el cual ejecutará la sentencia y nos devolverá el conjunto de resultados directamente.
- Es decir, no es necesario hacer la operación en 2 pasos (`prepare()` y `execute()`) como hacíamos hasta ahora.

```
$stmt = $dbh->query('
    SELECT nombre, apellidos, edad
    FROM empleado');

// Establecemos el modo en el que queremos
recibir los datos
$stmt->setFetchMode(PDO::FETCH_ASSOC);

while($row = $stmt->fetch()) {
    echo $row['nombre'] . "--";
    echo $row['apellidos'] . "--";
    echo $row['edad ' ] . "<br>";
}
```

fetchObject()

- Alternativa al método **fetch()** la cual devolverá los resultados como objetos anónimos (`PDO::FETCH_OBJ`) u objetos de la clase indicada (`PDO::FETCH_CLASS`).

```
$stmt = $dbh->query('
    SELECT nombre, apellidos, edad
    FROM empleado');

while($persona = $stmt
->fetchObject())
{
    echo $persona->nombre;
    echo $persona->apellido;
}
```

```
$stmt = $dbh->query('
    SELECT nombre, apellidos, edad
    FROM empleado');

while($persona = $stmt
->fetchObject('Alumno')) {
    echo $persona->nombre;
    echo $persona->apellido;
}
```

fetchAll()

- A diferencia del método `fetch()`, `fetchAll()` trae todos los datos de golpe, sin abrir ningún puntero, almacenándolos en un array.
- Se recomienda cuando no se esperan demasiados resultados que podrían provocar problemas de memoria.

```
// $resultado contendrá un array asociativo con todos los datos
$resultado = $stmt->fetchAll(PDO::FETCH_ASSOC);

// Para leer las filas podemos recorrer el array y acceder a la
información.
foreach ($resultado as $row){
    echo $row["nombre"]." ".$row["apellido"].PHP_EOL;
}
```

Hands on!

01. Lista de la compra: crea una aplicación que muestre una lista de la compra almacenada en base de datos. La tabla de base de datos únicamente tendrá dos columnas, una con el ID y otra un VARCHAR con el texto (será el nombre del elemento a comprar).

- Añade a la aplicación anterior un formulario para introducir nuevos productos en la lista.
- Añade a la aplicación anterior un enlace a cada producto de la lista para que se pueda eliminar de la lista.

Lista de compra

- Pan ([Eliminar](#))
- Leche ([Eliminar](#))
- Huevos ([Eliminar](#))
- Cereales ([Eliminar](#))
- Fruta ([Eliminar](#))

Añadir elemento

[Vaciar lista](#)

Hands on!

02. Crea una aplicación para el mantenimiento de empleados de la empresa. La aplicación deberá tener las siguientes opciones:

- Mostrar todos los empleados en una tabla.
- Insertar un empleado.
- Eliminar un empleado.
- Mostrar detalle de un empleado (abrirá una página aparte).

App de Gestión de Empleados

Bienvenido a la aplicación de aprendizaje Gestión de Empleados. Este ejercicio tiene como objetivo repasar el acceso a datos mediante PDO y comenzar a separar la lógica de las páginas de la presentación y del acceso a datos.

Listado de empleados

| DNI | Nombre | Apellidos | Opciones |
|-----------|--------|-----------------|---|
| 72752343Z | Ane | Urrutia Larrain | Ver detalles Eliminar |
| 75854342A | Mikel | Abasolo Lerena | Ver detalles Eliminar |
| 76852372Z | Aritz | Boroa Zabalburu | Ver detalles Eliminar |
| 12345678H | June | Fernández | Ver detalles Eliminar |

* Opción secreta: **Vaciar lista**

Añadir nuevo empleado

Nombre

Apellidos

Edad

01/01/2000

Email

DNI

Mujer

Curriculum

Añadir

App de Gestión de Empleados

Bienvenido a la aplicación de aprendizaje Gestión de Empleados. Este ejercicio tiene como objetivo repasar el acceso a datos mediante PDO y comenzar a separar la lógica de las páginas de la presentación y del acceso a datos.

| | |
|---------------------|---|
| DNI | 72752343Z |
| Nombre | Ane |
| Apellidos | Urrutia Larraín |
| Edad | 27 |
| Sexo | Mujer |
| Fecha de nacimiento | 1992-10-06 |
| Curriculum | Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec varius tellus turpis, et cursus urna aliquet non. Aenean in mi mattis, bibendum est sed, accumsan metus. Ut dapibus rhoncus sem, eu condimentum turpis hendrerit dignissim. Integer accumsan mauris tellus, in fermentum ex dignissim in. Vestibulum quis fringilla sapien, at tempus. |

Volver

Hands on!

03. AMPLIACIÓN DEL EJERCICIO 02: añade un buscador a la aplicación anterior. El usuario podrá introducir el nombre del empleado y mostrar en la tabla únicamente los empleados cuyo nombre coincida exactamente con el nombre introducido por el usuario. Podrá volver a ver todos los registros buscando un texto vacío.

Listado de empleados

| <input type="text" value="Introduce el nombre exacto"/> | | | <button>filtrar</button> |
|---|--------|----------------|---|
| DNI | Nombre | Apellidos | Opciones |
| 75854342A | Mikel | Abasolo Lerena | Ver detalles Eliminar |
| 71734321O | Mikel | López Madina | Ver detalles Eliminar |

App de Gestión de Empleados

Bienvenido a la aplicación de aprendizaje Gestión de Empleados. Este ejercicio tiene como objetivo repasar el acceso a datos mediante PDO y comenzar a separar la lógica de las páginas de la presentación y del acceso a datos.

Listado de empleados

| DNI | Nombre | Apellidos | Opciones |
|-----------|--------|----------------|---|
| 75854342A | Mikel | Abasolo Lerena | Ver detalles Eliminar |
| 71734321O | Mikel | López Madina | Ver detalles Eliminar |

* Opción secreta: [Vaciar lista](#)

Añadir nuevo empleado

Sources

- Github jvadillo: <https://github.com/jvadillo/guia-php-pdo>
- WikiBooks PHP: https://en.wikibooks.org/wiki/PHP_Programming