UNIVERSIDAD TECNOLÓGICA DE PANAMÁ

Facultad de Ingeniería de Sistemas Computacionales

Desarrollo de Software VII (PHP)

Kexy Rodríguez

11 de octubre de 2018

Contenido

- Conceptos
- Conexión
- Consultas

Conceptos – Base de datos

“La extensión Objetos de Datos de PHP (PDO por sus siglás en inglés) define una interfaz ligera para poder acceder a bases de datos en PHP. Cada controlador de bases de datos que implemente la interfaz PDO puede exponer características específicas de la base de datos, como las funciones habituales de la extensión”.

“PDO proporciona una capa de abstracción de acceso a datos, lo que significa que, independientemente de la base de datos que se esté utilizando, se emplean las mismas funciones para realizar consultas y obtener datos. PDO no proporciona una abstracción de bases de datos; no reescribe SQL ni emula características ausentes. Se debería usar una capa de abstracción totalmente desarrollada si fuera necesaria tal capacidad”.

Fuente: http://php.net/manual/es/intro.pdo.php (PHP.Net, 2018)

Conexión – ejemplo 1

```php
<?php
try {
$dsn = "mysql:host=localhost;dbname=$baseDeDatos";
$conexion = new PDO($dsn, $usuario, $contraseña);
} catch (PDOException $e){
echo $e->getMessage();
}
?>
```

Conexión – ejemplo 2

```php
// Con un array de opciones
try {
$dsn = "mysql:host=localhost;dbname=$dbname";
$options = array(
PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION
);
$conexion = new PDO($dsn, $user, $password);
} catch (PDOException $e){
echo $e->getMessage();
}
// Con un el método PDO::setAttribute
try {
$dsn = "mysql:host=localhost;dbname=$dbname";
$conexion = new PDO($dsn, $user, $password);
$conexion ->setAttribute(PDO::ATTR_ERRMODE,
PDO::ERRMODE_EXCEPTION);
} catch (PDOException $e){
echo $e->getMessage();
}
```

PDO maneja los errores en forma de excepciones, por lo que la conexión siempre ha de ir encerrada en un bloque try/catch. Se puede (y se debe) especificar el modo de error estableciendo el atributo *error mode*:

Consulta – inserción ejemplo 1

Cuando se **obtienen, insertan o actualizan datos**, el esquema es: **PREPARE -> [BIND] -> EXECUTE**.
Se pueden indicar los parámetros en la sentencia con un interrogante "**?**" o mediante un **nombre específico**.

```php
// Prepare
$consulta = $conexion->prepare("INSERT INTO Clientes (nombre, apellido) VALUES (?, ?)");
// Bind
$nombre = "Juan";
$apellido = "Alveo";
$consulta->bindParam(1, $nombre);
$consulta->bindParam(2, $apellido);
// Excecute
$consulta->execute();
```

Consulta – inserción ejemplo 2

```php
// Prepare
$consulta = $conexion->prepare("INSERT INTO Clientes (nombre, apellido) VALUES (:nombre, :apellido)");
// Bind
$nombre = "Juan";
$apellido = "Alveo";
$consulta->bindParam(':nombre', $nombre);
$consulta->bindParam(':apellido', $apellido);
// Excecute
$consulta->execute();
```

Existen dos métodos para enlazar valores: *bindParam()* y *bindValue()*:

En la práctica *bindValue()* se suele usar cuando se tienen que insertar datos sólo una vez, y *bindParam()* cuando se tienen que pasar datos múltiples (desde un array por ejemplo).

http://php.net/manual/es/pdostatement.bindparam.php

Consulta – inserción ejemplo 3

También existe un **método lazy**, que es **pasando los valores mediante un array** (siempre array, aunque sólo haya un valor) al método *execute()*:

```
// Prepare:
$consulta = $conexion->prepare("INSERT INTO Clientes (nombre, apellido) VALUES (:nombre, :apellido)");
$nombre = "Juana";
$apellido = "Ramos";
// Bind y execute:
if($consulta->execute(array(':nombre'=>$nombre, ':apellido'=>$apellido))) {
echo "Se ha insertado un nuevo registro!";
}
```

Consulta – inserción ejemplo 3

También existe un **método lazy**, que es **pasando los valores mediante un array** (siempre array, aunque sólo haya un valor) al método *execute()*:

```
// Prepare:
$consulta = $conexion->prepare("INSERT INTO Clientes (nombre, apellido) VALUES (:nombre, :apellido)");
$nombre = "Juana";
$apellido = "Ramos";
// Bind y execute:
if($consulta->execute(array(':nombre'=>$nombre, ':apellido'=>$apellido))) {
echo "Se ha insertado un nuevo registro!";
}
```

Consulta de datos ejemplo 1

```php
// FETCH_ASSOC
$consulta = $conexion->prepare("SELECT * FROM Usuario");
// Especificamos el fetch mode antes de llamar a fetch()
$consulta->setFetchMode(PDO::FETCH_ASSOC);
// Ejecutamos
$consulta->execute();
// Mostramos los resultados
while ($row = $consulta->fetch()){
echo "Nombre: {$row["nombre"]} <br>";
echo “Apellido: {$row["apellido"]} <br><br>";
}
```

PDO::FETCH_ASSOC: devuelve un array indexado cuyos keys son el nombre de las columnas.

Consulta de datos ejemplo 2

```php
// FETCH_OBJ
$consulta = $conexion->prepare("SELECT * FROM usuario");
// Ejecutamos
$stmt->execute();
// Ahora vamos a indicar el fetch mode cuando llamamos a fetch:
while($row = $consulta->fetch(PDO::FETCH_OBJ)){
echo "Nombre: " . $row->nombre . "<br>";
echo “Apellido: " . $row->apellido . "<br>";
}
```

PDO::FETCH_OBJ: devuelve un objeto anónimo con nombres de propiedades que corresponden a las columnas.

Consulta de datos ejemplo 3

```php
// FETCH_OBJ
$consulta = $conexion->prepare("SELECT * FROM usuario");
// Ejecutamos
$stmt->execute();
// Ahora vamos a indicar el fetch mode cuando llamamos a fetch:
while($row = $consulta->fetch(PDO::FETCH_OBJ)){
echo "Nombre: " . $row->nombre . "<br>";
echo “Apellido: " . $row->apellido . "<br>";
}
```

PDO::FETCH_OBJ: devuelve un objeto anónimo con nombres de propiedades que corresponden a las columnas.

Consulta de datos ejemplo 3

PDO::FETCH_ASSOC: devuelve un array indexado cuyos keys son el nombre de las columnas.
PDO::FETCH_NUM: devuelve un array indexado cuyos keys son números.
PDO::FETCH_BOTH: valor por defecto. Devuelve un array indexado cuyos keys son tanto el nombre de las columnas como números.
PDO::FETCH_BOUND: asigna los valores de las columnas a las variables establecidas con el método PDOStatement::bindColumn.
PDO::FETCH_CLASS: asigna los valores de las columnas a propiedades de una clase. Creará las propiedades si éstas no existen.
PDO::FETCH_INTO: actualiza una instancia existente de una clase.
PDO::FETCH_OBJ: devuelve un objeto anónimo con nombres de propiedades que corresponden a las columnas.
PDO::FETCH_LAZY: combina PDO::FETCH_BOTH y PDO::FETCH_OBJ, creando los nombres de las propiedades del objeto tal como se accedieron.