

# PDO en profundidad

**Clase 11. Operaciones comunes**

Carlos Ruiz Ruso • @micromante

# Repaso rápido

**Pequeño repaso de la Clase 10**



# Prepare() vs Query()

- **Query()**
  - Cuando usamos `$db->query('SELECT * FROM table WHERE id = 1');`
  - La query se lanza sin escapar los parametros
  - No podemos reutilizar las consultas
- **Prepare()**
  - Cuando usamos `$q = $db->prepare('SELECT * FROM table WHERE id = :id');`
  - La query se lanzará al ejecutar `$q->execute()` no antes
  - Podemos usar la plantilla `$q` en otros momentos
  - Podemos enlazar parametros con `bindParam` o `bindValue`

## bindParam vs bindValue

Ejemplo \$q = \$db->prepare('SELECT \* FROM table WHERE id = :id');

- **bindParam()**
  - Si hacemos el enlace \$q->bindParam(\$value) asociamos el parámetro o variable, el valor puede cambiar antes de hacer el execute()
- **bindValue()**
  - Si hacemos el enlace \$q->bindValue(\$value) asociamos el valor de la variable a prepare, el valor NO puede cambiar ya porque esta evaluado



# Contenidos de la clase PDO

- PDO::beginTransaction – Inicia una transacción
- PDO::commit – Consigna una transacción
- PDO::errorInfo – Obtiene información extendida del error asociado con la última operación del manejador de la base de datos
- PDO::exec – Ejecuta una sentencia SQL y devuelve el número de filas afectadas
- PDO::getAttribute – Devuelve un atributo de la conexión a la base de datos
- PDO::inTransaction – Comprueba si una transacción está activa
- PDO::lastInsertId – Devuelve el ID de la última fila o secuencia insertada
- PDO::prepare – Prepara una sentencia para su ejecución y devuelve un objeto sentencia
- PDO::query – Ejecuta una sentencia SQL, devolviendo un conjunto de resultados como un objeto PDOStatement
- PDO::quote – Entrecomilla una cadena de caracteres para usarla en una consulta
- PDO::rollBack – Revierte una transacción
- PDO::setAttribute – Establece un atributo

## Transacciones y 'auto-commit'

- Una vez tenemos el objeto PDO
- Sea de golpe o por etapas, se garantiza que se aplicaran los cambios a la base de datos de forma segura y sin interferencia de otras conexiones una vez realizamos el 'commit'
- El trabajo puede ser deshecho automáticamente bajo petición
- **Lotes** de operaciones a la vez con prepare, query o exec
- Debemos abrir la conexión con **PDO::beginTransaction**
- Podemos usar metodos como **PDO::commit()** y **PDO::rollBack()**
- **Desactiva el modo autocommit**



## Ejemplo inicio de transacción

...

```
$conn = new PDO(...);
```

```
$conn->beginTransaction(); //Deshabilitamos el autocommit para  
operaciones que no sean del tipo drop
```

```
$conn->exec('query 1');
```

```
$conn->exec('query 2'); ... //Podemos enlazar mas operaciones
```

```
$conn->commit();
```

...

## Ejemplo inicio de transacción con prepare()

```
$cnx = new PDO($dsn,$dbuser,$dbpass);  
$cnx->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);  
$cnx->beginTransaction();  
$stmt= $cnx->prepare("SELECT * FROM users WHERE username=?");  
$stmt->execute(array($user_input));  
$stmt_2= $cnx->prepare("SELECT * FROM othertable WHERE  
some_column=?");  
$stmt_2->execute(array($user_input_2));  
$cnx->commit(); //No se ejecuta hasta aqui
```



## Si falla el commit, rollBack()

...

```
} catch (Exception $e) {
```

```
    //Se produce un fallo o excepción, revertimos
```

```
    $conn->rollBack();
```

```
    echo 'Error: '. $e->getMessage();
```

```
}
```

...

## lastInsertId()

Muchas bases de datos tienen un **único identificador por fila**, podemos acceder al último valor insertado.

```
if($db->exec("INSERT INTO...")) { //Devuelve 1 si fila afectada
    $id = $db->lastInsertId(); //Comprobamos el ultimo id
}
```



## getAttribute() I

Mediante el método `getAttribute` podemos consultar la información del estado de la base de datos.

```
//Versión de la base de datos
```

```
echo $db->getAttribute(PDO::ATTR_SERVER_VERSION);
```

```
//Versión de la librería cliente
```

```
echo $db->getAttribute(PDO::ATTR_CLIENT_VERSION);
```

## getAttribute() II

Información del servidor y del estado de la conexión.

//Info server

```
echo $db->getAttribute(PDO::ATTR_SERVER_INFO);
```

//Estado de la conexión

```
echo $db->getAttribute(PDO::ATTR_CONNECTION_STATUS);
```



## Diferencias entre `exec()` y `query()`

- **Exec()** ejecuta una consulta SQL y devuelve las **filas afectadas** no es **PDOStatement**
- **Query()** ejecuta una consulta SQL y devuelve un **conjunto de resultados** en un objeto **PDOStatement**

## Seguridad con PDO::quote

Siempre debemos comprobar el contenido de las variables que vamos a lanzar sobre la base de datos y escapar posibles inyecciones de código.

**Con quote podemos escapar comillas y operadores extraños.**

```
$cadena = 'Ejemplo \' peligroso';  
echo $conn->quote($cadena);
```

//Devuelve: 'Ejemplo " peligroso'



## Repaso de PDO::errorInfo

Obtiene información del error sobre la última operación, con esto podemos corregir posibles errores y cambiar el flujo del programa.

Devuelve array con 3 campos

[0 => Código error, identificador, 1=> Código del driver, 2 => Mensaje]

```
print_r( $db->errorInfo() );
```

# Contenidos de la clase PDOStatement

- PDOStatement::bindParam — Vincula un parámetro al nombre de variable especificado
- PDOStatement::bindValue — Vincula un valor a un parámetro
- PDOStatement::execute — Ejecuta una sentencia preparada
- PDOStatement::fetch — Obtiene la siguiente fila de un conjunto de resultados
- PDOStatement::fetchAll — Devuelve un array que contiene todas las filas del conjunto de resultados
- PDOStatement::fetchObject — Obtiene la siguiente fila y la devuelve como un objeto
- PDOStatement::getAttribute — Recupera un atributo de sentencia
- PDOStatement::getColumnMeta — Devuelve metadatos de una columna de un conjunto de resultados
- PDOStatement::rowCount — Devuelve el número de filas afectadas por la última sentencia SQL



## PDOStatement::fetchAll

Obtiene un array con todas las filas del conjunto de resultados, manera recomendable aunque debemos llevar cuidado cuando son mucho resultados por el límite de las variables de PHP.

```
$conn->fetchAll(PDO::FETCH_CLASS);  
Devuelve como un objeto
```

## PDOStatement::fetch

Obtiene la siguiente fila del conjunto de resultados, en el bucle debemos llamarlo cada vez para obtener el resultado siguiente.

```
$conn->fetch(PDO::FETCH_ASSOC);  
Devuelve como array
```



## fetchAll(PDO:format) vs fetch(PDO:format)

```
$conn->fetchAll(PDO::FETCH_CLASS);
```

//Todas las filas como objeto

```
$conn->fetch(PDO::FETCH_CLASS);
```

//Fila por fila, debemos llamarlo en bucle para obtener la siguiente, el formato tambien objeto

## PDOStatement::getColumnMeta

En ocasiones necesitamos obtener los metadatos que tiene la tabla, estos podrían ser el nombre, descripción, tamaño del campo, etc...

```
$conn->getColumnMeta($i);
```



# EJEMPLO CREACIÓN CLASE BASE PDO

## PDO WRAPPER LIBRARY

<https://github.com/salebab/database>



# EASY CRUD Library

<https://github.com/indieteq/PHP-MySQL-PDO-Database-Class>

# MUCHAS GRACIAS A TODOS!

Podeis seguirme en la redes sociales como @micromante o Carlos Ruiz Ruso  
[www.micromante.com](http://www.micromante.com)