UCH UNIVERSIDAD DE CIENCIAS Y HUMANIDADES

PROGRAMACIÓN II

SEMANA 13 - 2
PHP DATA OBJECT - CONSULTAS

Eric Gustavo Coronel Castillo
ecoronel@uch.edu.pe
gcoronelc.blogspot.com

PHP Data Object - Consultas

Las consultas a una base de datos es las operación más ejecutada, es por eso importante conocer todas las posibilidades que te brinda PDO para elegir la mejor de ellas.

Índice

1 Consultas Sin Parámetros

Para ejecutar consultas sin parámetros tenemos dos opciones:

1. **Ejecución Directa:** Utilizar el método **query** de la clase PDO para ejecutar directamente la sentencia SELECT.
2. **Ejecución con Procedimiento Almacenado:** La consulta la ubicamos dentro de un procedimiento almacenado y luego utilizamos el método **query** de la clase PDO para ejecutarlo.

1.1 Ejecución Directa

En este caso utilizamos el método **query** de la clase PDO para ejecutar directamente la sentencia SELECT

Ejemplo 1

El siguiente ejemplo muestra los movimiento de la cuenta **00100001**, note que el código de la cuenta esta fijo, no hay manera de cambiarlo, por lo tanto se trata de una consulta sin parámetros.

```
<?php

try {
   $dsn = 'mysql:host=localhost;dbname=eurekabank';
   $dbh = new PDO($dsn, 'root', 'mysql');
   $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   $dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
   $dbh->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
   $query = "select * from movimiento where chr_cuencodigo = '00100001'";
   $stm = $dbh->query($query);
   while( $row = $stm->fetch() ){
      echo "{$row['dtt_movifecha']} - {$row['chr_tipocodigo']} –
         {$row['dec_moviimporte']}<br/>";
   }
   $dbh = null;
} catch( PDOException $e ) {
   echo $e->getMessage();
}

?>
```

El resultado de su ejecución es el siguiente:

```
2008-01-06 - 001 - 2800.00
2008-01-15 - 003 - 3200.00
2008-01-20 - 004 - 800.00
2008-02-14 - 003 - 2000.00
2008-02-25 - 004 - 500.00
2008-03-03 - 004 - 800.00
2008-03-15 - 003 - 1000.00
```

1.2 Ejecución con Procedimiento Almacenado

Ejemplo 2

El siguiente ejemplo ilustra como ejecutar un procedimiento almacenado que no tiene parámetros.

A continuación tenemos el procedimiento almacenado para consultar de todas las cuentas.

```
DELIMITER $$

DROP PROCEDURE IF EXISTS usp_consulta_cuentas$$

CREATE PROCEDURE usp_consulta_cuentas()
BEGIN
      select * from cuenta;
END$$

DELIMITER ;
```

Y el siguiente programa muestra el saldo de todas las cuentas.

```
<?php

try {
   $dsn = 'mysql:host=localhost;dbname=eurekabank';
   $dbh = new PDO($dsn, 'root', 'mysql');
   $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   $dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
   $dbh->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
   $query = "CALL usp_consulta_cuentas()";
   $stm = $dbh->query($query);
   while( $row = $stm->fetch() ) {
      echo "{$row['chr_cuencodigo']} - {$row['dec_cuensaldo']} <br/>";
   }
   $dbh = null;
} catch( PDOException $e ) {
   echo $e->getMessage();
```

```
}

?>
```

El resultado de la ejecución es el siguiente:

```
00100001 - 6900.00
00100002 - 4500.00
00200001 - 7000.00
00200002 - 6800.00
00200003 - 6000.00
```

2 Consultas con Parámetros

Para ejecutar una consulta con parámetros tenemos dos alternativas:

1. **Sentencias Preparadas.-** Las sentencias preparadas permiten definir parámetros y asignarle sus respectivos valores antes de ejecutarlas.
2. **Procedimientos Almacenados.-** Los procedimientos almacenados también permiten definir parámetros que podemos utilizarlos cuando ejecutamos una consulta.

El uso de sentencias preparada elimina la posibilidad de violar la seguridad utilizando SQL Injection.

2.1 Sentencias Preparadas

Para preparar una sentencia SQL debemos ejecutar el método **prepare** de la clase PDO, a continuación tenemos su sintaxis:

```
PDOStatement prepare ( string $statement [, array $driver_options = array() ] )
```

Este método retorna un objeto de tipo PDOStatement.

La sentencia de SQL puede contener cero o más parámetros con nombre (:nombre) o signo de interrogación (?).

El siguiente script muestra cómo preparar una sentencia con un parámetro con signo de interrogación.

```
$stm = $dbh->prepare('select * from cliente chr_cliecodigo = ?');
```

El siguiente script muestra cómo preparar una sentencia con un parámetro con nombre.

```
$stm = $dbh->prepare('select * from cliente chr_cliecodigo = :codigo');
```

Los parámetros serán sustituidos por sus valores reales cuando se ejecuta la instrucción. No se puede utilizar al mismo tiempo los nombres y signos de interrogación para definir parámetros dentro de la misma sentencia SQL; se debe escoger un solo estilo.

Para ejecutar una sentencia preparada debemos utilizar el método **execute** de la clase **PDOStatement**. Si la sentencia preparada tiene, tenemos las siguientes alternativas:

- Utilizar el método **bindParam** de la clase **PDOStatement** para ligar a variables PHP a un parámetro. Una variable ligada pasar su valor como entrada y recibir el valor de salida, de su parámetro asociado. Esta operación se realiza con cada uno de los parámetros.

- Pasar una matriz de valores solo para los parámetros de entrada.

Los parámetros con nombre deben ser únicos para cada valor que desea pasar a la sentencia cuando esta es ejecutada con el método **execute** de la clase **PDOStatement**. No se puede utilizar el mismo nombre de parámetro dos veces en una declaración preparada.

Otra característica de una sentencia preparada es que podemos ejecutarla varias veces pero con diferentes valores de los parámetros y de esta manera optimiza el rendimiento de la aplicación.

Ejemplo 3

El siguiente ejemplo muestra cómo utilizar el método **execute** de la clase **PDOStatement** utilizando un arreglo para pasar el valor del parámetro.

```
<?php

try {
   $dsn = 'mysql:host=localhost;dbname=eurekabank';
   $dbh = new PDO($dsn, 'root', 'mysql');
   $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   $dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
   $dbh->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
   $query = "select * from cliente where chr_cliecodigo = ?";
   $stm = $dbh->prepare($query);
   $stm->execute( array('00013') );
   if( $stm->rowCount() == 0){
      throw new PDOException( 'Código no existe.' );
   }
   $row = $stm->fetch();
   foreach ($row as $column => $value){
      echo "$column = $value<br/>";
   }
   $dbh = null;
} catch( PDOException $e ) {
   echo $e->getMessage();
}

?>
```

El resultado es el siguiente:

```
chr_cliecodigo = 00013
vch_cliepaterno = RICALDE
vch_cliematerno = RAMIREZ
vch_clienombre = ROSARIO ESMERALDA
chr_cliedni = 10761324
vch_clieciudad = LIMA
vch_cliedireccion = MIRAFLORES
vch_clietelefono = 991-23546
vch_clieemail = r.ricalde@gmail.com
```

Si queremos utilizar el método **bindParam** debemos remplazar la sentencia:

```
$stm->execute( array('00013') );
```

Por estas otras:

```
$codigo = '00013';
$stm->bindParam(1,$codigo,PDO::PARAM_STR, 5);
$stm->execute();
```

Tenga en cuenta que cuando utilizamos **bindParam** se ligan variables a los parámetros, por eso es necesario crear la variable **$codigo**.

Ejemplo 4

En este ejemplo ilustramos el uso de parámetro con nombre, se trata de consultar los movimientos de una cuenta.

Debemos crear un arreglo asociativo al momento de pasar el dato al parámetro **:codigo** en el método **execute**.

Note también que se recuperan todas las filas utilizando el método **fetchAll** en un arreglo de nombre **$lista**, cada elemento de este arreglo representa una fila de la consulta como un arreglo asociativo donde los índices son los nombres de las columnas en minúsculas.

```
<?php

try {
   $dsn = 'mysql:host=localhost;dbname=eurekabank';
   $dbh = new PDO($dsn, 'root', 'mysql');
   $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   $dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
   $dbh->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
   $query = "select * from movimiento where chr_cuencodigo = :codigo";
   $stm = $dbh->prepare($query);
   $stm->execute( array(':codigo'=>'00100001') );
   if( $stm->rowCount() == 0){
      throw new PDOException( 'Cuenta no existe.' );
   }
   $lista = $stm->fetchAll();
   foreach ($lista as $row){
      echo "{$row['dtt_movifecha']} - {$row['chr_tipocodigo']} -
{$row['dec_moviimporte']}<br/>";
   }
   $dbh = null;
} catch( PDOException $e ) {
   echo $e->getMessage();
}

?>
```

El resultado es el siguiente:

```
2008-01-06 - 001 - 2800.00
2008-01-15 - 003 - 3200.00
2008-01-20 - 004 - 800.00
2008-02-14 - 003 - 2000.00
2008-02-25 - 004 - 500.00
2008-03-03 - 004 - 800.00
2008-03-15 - 003 - 1000.00
```

Si queremos utilizar el método **bindParam** debemos remplazar la sentencia:

```
$stm->execute( array(':codigo'=>'00100001') );
```

Por estas otras:

```
$codigo = '00100001';
$stm->bindParam(':codigo',$codigo,PDO::PARAM_STR);
$stm->execute();
```

Tenga en cuenta que cuando utilizamos **bindParam** se ligan variables a los parámetros, por eso es necesario crear la variable **$codigo**.

2.2 Procedimientos Almacenados

Los procedimientos almacenados también permiten definir parámetros que podemos utilizarlos cuando ejecutamos una consulta.

Ejemplo 5

Se necesita consultar el saldo total para una determinada sucursal, a continuación tenemos el procedimiento almacenado para tal propósito:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS usp_consultar_saldo_total$$

CREATE PROCEDURE usp_consultar_saldo_total
(IN p_sucursal char(3))
BEGIN

	select sum(dec_cuensaldo) as saldo_total
	from cuenta
	where chr_sucucodigo = p_sucursal;

END$$

DELIMITER ;
```

En este ejemplo el procedimiento retorna un result set conteniendo solamente una fila con una columna.

La columna del result set que retorna el procedimiento almacenado es vinculada con la variable **$saldo_total**, esta variable va a tomar el de la fila actual cuando ejecute la función **fetch** de la clase PDOStatement.

```
<?php

try {
   $dsn = 'mysql:host=localhost;dbname=eurekabank';
   $dbh = new PDO($dsn, 'root', 'mysql');
   $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   $dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
   $dbh->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
   $query = "CALL usp_consultar_saldo_total(?)";
   $stm = $dbh->prepare($query);
   $sucursal = '001';
   // Parámetro de Entrada
   $stm->bindParam(1,$sucursal,PDO::PARAM_STR,3);
   // Vinculando una variable a la columna de la salida
   $stm->bindColumn(1,$saldo_total);
```

```
    $stm->execute();
    $stm->fetch();
    $dbh = null;
    echo "Saldo Total: $saldo_total";
} catch( PDOException $e ) {
    echo $e->getMessage();
}

?>
```

El resultado que se obtiene es el siguiente:

Saldo Total: 11400.00

Ejemplo 6

Se necesita consultar los movimientos de una determinada cuenta, a continuación tenemos el procedimiento almacenado para tal propósito:

```
DELIMITER $$

DROP PROCEDURE IF EXISTS usp_consultar_movimientos$$

CREATE PROCEDURE usp_consultar_movimientos
(IN p_cuenta char(8))
BEGIN

      select *
      from movimiento
      where chr_cuencodigo = p_cuenta;

END$$

DELIMITER ;
```

En este ejemplo el procedimiento almacenado retorna un result set con conformado por varias filas y columnas.

Las columnas del result set se vinculan con variables para que asigne los valores automáticamente cada vez que se ejecuta la función **fetch**, el programa es el siguiente:

```
<?php

try {
   $dsn = 'mysql:host=localhost;dbname=eurekabank';
   $dbh = new PDO($dsn, 'root', 'mysql');
   $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
   $dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
   $dbh->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
   $query = "CALL usp_consultar_movimientos(?)";
   $stm = $dbh->prepare($query);
   $cuenta = '00100001';
   // Parámetro de Entrada
   $stm->bindParam(1,$cuenta,PDO::PARAM_STR,8);
   // Vinculando columnas del result set con variables
   $stm->bindColumn('dtt_movifecha',$fecha);
   $stm->bindColumn('chr_tipocodigo',$tipo);
   $stm->bindColumn('dec_moviimporte',$importe);
   $stm->execute();
   while( $stm->fetch() ) {
      echo "$fecha - $tipo - $importe <br/>";
   }
   $dbh = null;
```

```
} catch( PDOException $e ) {
   echo $e->getMessage();
}

?>
```

El resultado es el siguiente:

```
2008-01-06 - 001 - 2800.00
2008-01-15 - 003 - 3200.00
2008-01-20 - 004 - 800.00
2008-02-14 - 003 - 2000.00
2008-02-25 - 004 - 500.00
2008-03-03 - 004 - 800.00
2008-03-15 - 003 - 1000.00
```