UCH UNIVERSIDAD DE CIENCIAS Y HUMANIDADES

PROGRAMACIÓN II

SEMANA 12

PHP DATA OBJECT

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

PHP Data Object

El mayor problema cuando desarrollados una aplicación con PHP es cuando nos cambian de motor de base de datos, salvo que apliquemos el patrón de diseño DAO con Abstract Factory y cada implementación del DAO sería para un motor diferente, pero aún así se trata de librerías diferentes con sus propios nombres de funciones y parámetros, y tener que lidiar con el SQL malicioso (SQL Injection).

PHP Data Object (PDO) es una extensión nativa de PHP5, provee una interface uniforme de acceso a datos, que nos permite mediante varios drivers, conectarnos a diferentes bases de datos.

Índice

1 Introducción

El mayor problema cuando desarrollados una aplicación con PHP es cuando nos cambian de motor de base de datos, salvo que apliquemos el patrón de diseño DAO con Abstract Factory y cada implementación del DAO sería para un motor diferente, pero aun así se trata de librerías diferentes con sus propios nombres de funciones y parámetros, y tener que lidiar con el SQL malicioso (SQL Injection).

Por ejemplo, cuando utilizamos las funciones de la librería de MySQL tenemos que preocuparnos por validar cada una de las variables que contienen la sentencia SQL que pasemos para su ejecución.

Primero tendríamos que verificar que **magic quotes** se encuentre desactivado, luego utilizando muchas funciones como por ejemplo **addslashes()** y **mysql_real_escape_string()** verificar que no quede ningún agujero en tu código.

En realidad esto es muy frustrante a la hora de ver que todo lo que has hecho para evitar las inyecciones SQL ha sido en vano, o tal vez después que te digan que no va en **MySQL** sino en **PostgreSQL** tendrías que cambiar tu código de nuevo.

PHP Data Object (PDO) es una extensión nativa de PHP5, provee una interface uniforme de acceso a datos, que nos permite mediante varios drivers, conectarnos a diferentes fuentes de datos.

Por ejemplo, la ventaja de utilizar el driver PDO de MySQL y no la librería de MySQL es que con este Driver es mucho más seguro trabajar con las sentencias SQL, ya que PDO se encarga de filtrar y evitar las inyecciones SQL.

2 Instalación y Configuración

Actualmente PHP tiene implementado drivers para varias fuentes de datos, la lista es la siguiente:

Driver name	Supported databases
PDO_DBLIB	FreeTDS / Microsoft SQL Server / Sybase
PDO_FIREBIRD	Firebird/Interbase 6
PDO_IBM	IBM DB2
PDO_INFORMIX	IBM Informix Dynamic Server
PDO_MYSQL	MySQL 3.x/4.x/5.x
PDO_OCI	Oracle Call Interface
PDO_ODBC	ODBC v3 (IBM DB2, unixODBC and win32 ODBC)
PDO_PGSQL	PostgreSQL
PDO_SQLITE	SQLite 3 and SQLite 2
PDO_4D	4D (Experimental)

Como puede apreciar tenemos drivers para las bases de datos más utilizadas en nuestro medio, así que no tenemos de que preocuparnos.

El driver para MySQL por defecto ya viene habilitado, si queremos utilizar otro motor debemos habilitar si driver en el archivo **PHP.INI**.

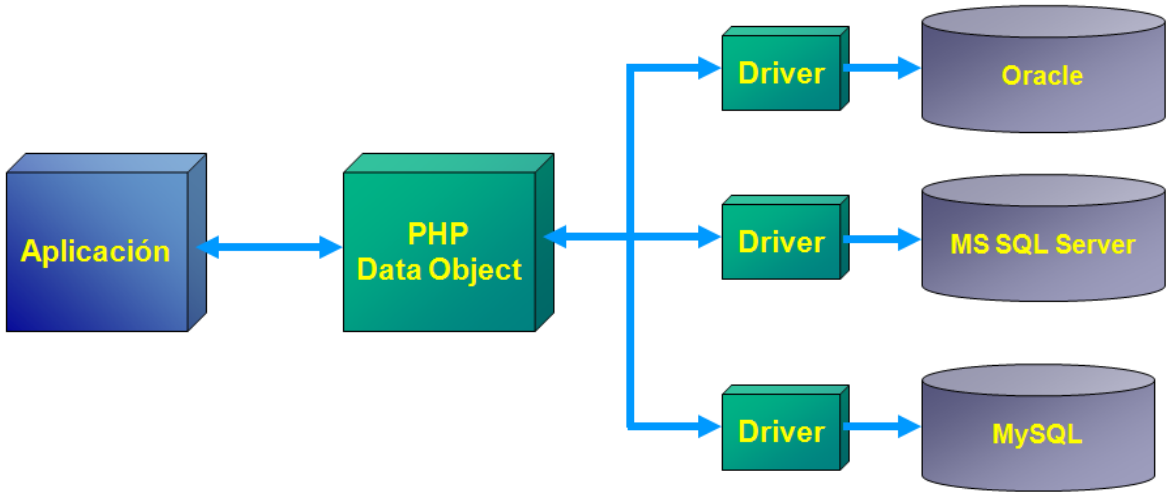

Figura 1 Esquema de trabajo con PHP PDO.

Por ejemplo, si queremos habilitar el driver para SQL Server buscamos la siguiente línea en el archivo PHP.INI.

```
;extension=php_pdo_mssql.dll
```

Lo que debemos hacer es quitarle el punto y coma (;) que se encuentra al inicio de la línea.

```
extension=php_pdo_mssql.dll
```

Y para que tenga efecto debemos reiniciar el servidor Apache. Para algunas ediciones de Linux debemos descargar el código fuente del y compilarlo.

Los proveedores de servicio de hosting debemos verificar que el driver PDO se encuentre habilitado antes de iniciar el proyecto.

3 Clases de la Extensión PDO

La extensión PHP_PDO está compuesta por tres clases:

Clase	Descripción
PDO	Representa una conexión entre PHP y un servidor de bases de datos.
PDOStatement	Representa una instrucción preparada y después que la instrucción es ejecutada, una result set.
PDOException	Representa un error lanzado por PDO.

3.1 La Clase PDO

Representa una conexión entre PHP y un servidor de bases de datos.

3.1.1 Estructura de la Clase

A continuación tenemos la estructura de la clase.

```
PDO {

    __construct ( string $dsn [, string $username
          [, string $password [, array $driver_options ]]] )
    bool beginTransaction ( void )
    bool commit ( void )
    mixed errorCode ( void )
    array errorInfo ( void )
    int exec ( string $statement )
    mixed getAttribute ( int $attribute )
    array getAvailableDrivers ( void )
    string lastInsertId ([ string $name = NULL ] )
    PDOStatement prepare ( string $statement
       [, array $driver_options = array() ] )
    PDOStatement query ( string $statement )
    string quote ( string $string
       [, int $parameter_type = PDO::PARAM_STR ] )
    bool rollBack ( void )
    bool setAttribute ( int $attribute , mixed $value )

}
```

3.1.2 Métodos

Los métodos son los siguientes:

Método	Descripción
__construct	Inicializa el objeto cuando creamos una instancia de la clase PDO, representa una conexión a una base de datos.
beginTransaction	Inicializa una transacción.
commit	Confirma una transacción.
errorCode	Recupera el **SQLSTATE** asociado con la última operación en la base de datos.
errorInfo	Recupera información extendida del error asociado con la última operación en la base de datos.
exec	Ejecuta una instrucción SQL y retorna el número de filas afectadas.
getAttribute	Recupera un atributo de conexión a base de dato.
getAvailableDrivers	Retorna un array (arreglo) de los drivers disponibles en la extensión PDO.
lastInsertId	Retorna el ID (identificador) de la última fila insertada o secuencia de valores.
prepare	Prepara una instrucción para ejecución y retorna un objeto de tipo **PDOStatement**.
query	Ejecuta una instrucción SQL, retornando un result set como un objeto de tipo **PDOStatement**.
quote	Quotes (Pone entre comillas simples un string) para uso en una query (consulta).
rollBack	Cancela una transacción.
setAttribute	Modifica un atributo.

Ejemplo 1

En este ejemplo ilustrare como hacer una conexión con MySQL.

```
<?php

   $dsn = 'mysql:dbname=eurekabank;host=localhost';
   $user = 'root';
   $password = 'mysql';

   try {
      $dbh = new PDO($dsn, $user, $password);
      echo 'Conexión ok.';
   } catch (PDOException $e) {
      echo 'Falló la conexión: ' . $e->getMessage();
   }

?>
```

Si no hay problemas tenemos el siguiente mensaje:

```
   Conexión ok.
```

Pero si por ejemplo la contraseña fuese incorrecta tenemos el siguiente mensaje:

```
Falló la conexión: SQLSTATE[28000] [1045] Access denied for user 'root'@'localhost'
(using password: YES)
```

3.2 La Clase PDOStatement

3.2.1 Estructura de la Clase

A continuación tenemos la estructura de la clase.

```
   PDOStatement implements Traversable {
      /* Propiedades */
      readonlystring $queryString;
      /* Métodos */
      bool bindColumn ( mixed $column , mixed &$param
         [, int $type [, int $maxlen [, mixed $driverdata ]]] )
      bool bindParam ( mixed $parameter , mixed &$variable
         [, int $data_type = PDO::PARAM_STR [, int $length
         [, mixed $driver_options ]]] )
      bool bindValue ( mixed $parameter , mixed $value
         [, int $data_type = PDO::PARAM_STR ] )
```

```
    bool closeCursor ( void )
    int columnCount ( void )
    bool debugDumpParams ( void )
    string errorCode ( void )
    array errorInfo ( void )
    bool execute ([ array $input_parameters = array() ] )
    mixed fetch ([ int $fetch_style = PDO::FETCH_BOTH
        [, int $cursor_orientation = PDO::FETCH_ORI_NEXT
        [, int $cursor_offset = 0 ]]] )
    array fetchAll ([ int $fetch_style = PDO::FETCH_BOTH
        [, int $column_index = 0 [, array $ctor_args = array() ]]] )
    string fetchColumn ([ int $column_number = 0 ] )
    mixed fetchObject ([ string $class_name = "stdClass" [, array $ctor_args ]] )
    mixed getAttribute ( int $attribute )
    array getColumnMeta ( int $column )
    bool nextRowset ( void )
    int rowCount ( void )
    bool setAttribute ( int $attribute , mixed $value )
    bool setFetchMode ( int $mode )
}
```

3.2.2 Métodos

Los métodos son los siguientes:

Método	Descripción
bindColumn	Enlazar una columna a una variable de PHP.
bindParam	Vincula un parámetro al nombre de la variable especificada.
bindValue	Liga un valor a un parámetro.
closeCursor	Cierra el cursor, lo que permite ejecutar la sentencia nuevamente.
columnCount	Retorna el número de columnas en el result set.
debugDumpParams	Muestra los datos contenidos en una sentencia preparada directamente en la salida.
errorCode	Recupera el SQLSTATE asociado con la última operación realizada en la base de datos.
errorInfo	Recupera información de error extendida asociada con la última operación realizada en la base de datos.
execute	Ejecuta una sentencia preparada.

fetch	Recupera la siguiente fina de un result set.
fetchAll	Retorna un arreglo que contiene todas las filas de un result set.
fetchColumn	Retorna una columna desde la siguiente fila de un result set.
fetchObject	Recupera la siguiente fila y la retorna como un objeto.
getAttribute	Recupera un atributo.
getColumnMeta	Recupera metadatos para una columna en un result set.
nextRowset	Avanza a la siguiente rowset en una sentencia que maneja multi-rowset.
rowCount	Retorna el número de filas afectadas por la última sentencia SQL.
setAttribute	Establece el valor de un atributo.
setFetchMode	Modifica el valor por defecto para recuperar datos de la base de datos para la instrucción actual.

Ejemplo 2

El siguiente ejemplo muestra una lista del nombre de todos los empleados.

```
<?php

$dsn = 'mysql:dbname=eurekabank;host=localhost';
$user = 'root';
$password = 'mysql';

try {
   $dbh = new PDO($dsn, $user, $password);
   $stm = $dbh->query("select * from empleado");
   while( $row = $stm->fetch() ){
      echo("{$row["vch_emplnombre"]}<br>");
   }
} catch (PDOException $e) {
   echo 'Error: ' . $e->getMessage();
}

?>
```

3.3 La Clase PDOException

3.3.1 Estructura de la Clase

A continuación tenemos la estructura de la clase.

```
PDOException extends RuntimeException {
    /* Propiedades */
    public array $errorInfo ;
    protected string $message ;
    protected string $code ;
    /* Métodos heredados */
    final public string Exception::getMessage ( void )
    final public Exception Exception::getPrevious ( void )
    final public int Exception::getCode ( void )
    final public string Exception::getFile ( void )
    final public int Exception::getLine ( void )
    final public array Exception::getTrace ( void )
    final public string Exception::getTraceAsString ( void )
    public string Exception::__toString ( void )
    final private void Exception::__clone ( void )
}
```

4 Conexión con la Base de Datos

4.1 Conexión

Las conexiones se establecen mediante la creación de instancias de la clase base **PDO**. No importa qué driver desea utilizar; siempre se utilizara la clase **PDO**. El constructor acepta parámetros para especificar la fuente de base de datos (conocido como el DSN) y, opcionalmente, el nombre de usuario y contraseña (si la hay).

La siguiente línea muestra un ejemplo de cómo sería la conexión con MySQL:

```
$dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
```

Los datos del usuario y la contraseña están en las variables **$user** y **$pass**.

Lo importante es que si hay algún error genera una excepción, por lo tanto debemos capturarla, a continuación tenemos un ejemplo:

```
try {

   $dsn = 'mysql:host=localhost;dbname=eurekabank';
   $dbh = new PDO($dsn, 'root', 'mysql');
   foreach($dbh->query('SELECT * from cliente') as $row) {
      echo("{$row['vch_clienombre']} <br/>");
   }
   $dbh = null;

} catch (PDOException $e) {

   echo("Error!: " . $e->getMessage() . "<br/>");

}
```

El script ilustra como mostrar los nombres de los clientes de la base de datos **eurekabank**, en caso de error un objeto **PDOException** será lanzado el cual es capturado en la sección **catch** y se muestra el respectivo mensaje.

Si la aplicación no captura la excepción lanzada desde el constructor **PDO**, la acción predeterminada por el motor Zend es terminar el script y mostrar un back trace. Este back trace probablemente revelará los detalles de conexión de base de datos completa, incluyendo el nombre de usuario y contraseña. Es su responsabilidad detectar esta excepción, ya sea explícitamente (a través de una declaración try - catch) o implícitamente a través de **set_exception_handler ()**.

Al conectarse con éxito a la base de datos, una instancia de la clase PDO se devuelve a su script. La conexión se mantiene activa durante toda la vida de ese objeto PDO. Para cerrar la conexión, es necesario destruir el objeto para garantizar que todas las referencias restantes sean eliminadas, esto se realiza asignándole NULL a la variable que contiene el objeto. Si no lo hace explícitamente, PHP cerrará automáticamente la conexión cuando el programa termina.

4.2 Estableciendo Atributos de la Conexión

Después de realizar la conexión con la base de datos es importante establecer los valores de algunos atributos para personalizar nuestra conexión.

4.2.1 Atributos PDO

El siguiente script muestra los atributos que acostumbro a establecer:

```
1. $dsn = 'mysql:host=localhost;dbname=eurekabank';
2. $dbh = new PDO($dsn, 'root', '');
3. $dbh->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
4. $dbh->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
5. $dbh->setAttribute(PDO::ATTR_CASE, PDO::CASE_LOWER);
```

En la línea 3 establezco el modo en que serán tratados los errores, en este caso le estoy indicando que debe generar una excepción.

En la línea 4 especifico que el método **fetch** devolverá cada fila como una matriz indexada por nombre de columna y como lo devuelve en el result set correspondiente. Si el result set contiene varias columnas con el mismo nombre, **PDO::FETCH_ASSOC** sólo devuelve un valor único por columna nombre. Esta característica también lo puedes establecer cuando ejecutas el **fetch**.

En la columna 5 fuerzo a que los nombres de las columnas se encuentren en minúsculas.

4.2.2 Atributos de la Base de Datos

En la mayoría de casos es necesario estableces algunos atributos de la base de datos, por ejemplo, cuando trabajamos con MySQL establecemos el atributo NAMES de la siguiente manera:

```
$dbh->query('SET NAMES UTF8');
```