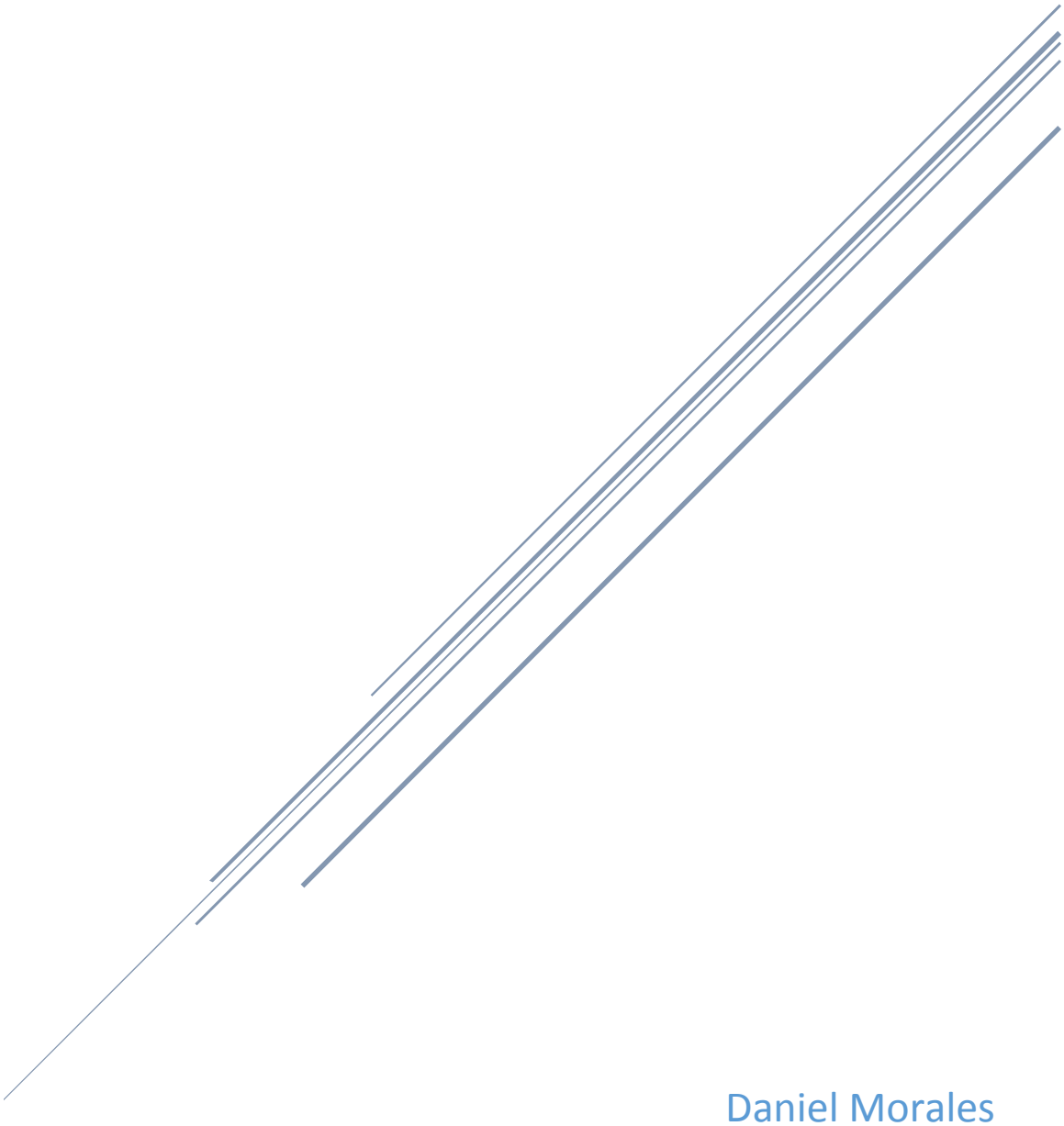
DOCUMENTACIÓN SQLML

Esta documentación está bajo revisión.



Daniel Morales
medennysoft@outlook.com

INTRODUCCIÓN

SQL methods Layer(SQLML) es una colección de clases que nos permiten de manera fácil y sencilla realizar consultas a la base de datos mediante el uso de simples métodos PHP. Utilizando objetos de datos PHP o por sus siglas en ingles PDO (PHP DATA OBJECTS). De este modo podemos asegurar un acceso seguro.

SQLML se basa en la ejecución de sentencias SQL utilizando métodos y paso de argumentos. Es así como se pretende facilitar la consulta, inserción, actualización y eliminación de los datos en la BD. Esto trae como beneficio la reducción de las líneas de código SQL, facilita la conexión mediante un método seguro al dbms, al utilizar PDO la conexión a diferentes DBMS es más sencillo, y más fácil de entender. Actualmente SQLML solo soporta 4 operaciones básicas pero se espera que en un futuro pueda soportar otras más.

CREANDO UN OBJETO SQLML

Primero que nada debemos de tener en cuenta que SQLML está programado en el paradigma orientado a objetos y como tal debe de seguir las reglas generales para la creación de objetos. Antes de comenzar a utilizar las clases del SQLML hay que incluir los ficheros en nuestro proyecto.

```
require_once __DIR__ . '/classes/Connection.php';

require_once __DIR__ . '/classes/SqlCollection.php';
```

Los archivos contienen dos clases, el archivo Connection.php contiene la clase “Connection”, mientras que el archivo sqlCollection contiene la clase “SqlCollection”. Ambas clases pertenecen al espacio de nombres “sqllayer”, por lo que al momento de hacer una instancia deberemos mencionar el espacio de nombres al que pertenece.

Una vez incluido los archivos en el proyecto entonces estaremos listos para crear el objeto SQLML, en este caso necesitamos simplemente realizar una instancia de la clase sqlCollection, ya que está última contiene todas las operaciones para establecer la conexión y ejecutar las sentencias.

La instancia de SqlCollection recibe opcionalmente como argumento un array que contendrá los datos de conexión de la BD. Es muy recomendable que se pase los datos al momento de crear el objeto.

Un ejemplo fácil y sencillo del array de configuración es la siguiente:

```
$cfg = array(
   'host' => 'localhost',
   'user' => 'root',
   'password' => 'root',
   'database' => 'db1_12312'
);
```

El DBMS predeterminado es MYSQL, si en caso de que usted utilice otro DBMS, tendrá que agregar la llave 'dbms' al array de configuración:

```
$cfg = array(
   'host' => 'localhost',
   'user' => 'root',
   'password' => 'root',
   'database' => 'db1_12312',
   'dbms' => 'mssql'
);
```

SI deseáramos crear una conexión sin especificar una base de datos posiblemente porque vayamos a utilizar DDL (lenguaje de definición de datos) entonces solo bastaría con omitir el elemento asociativo "database".

Por último en caso de querer definir una tabla default sobre las que se ejecutaran las consultas podemos hacer uso de la llave 'table_default' => 'tabla'. Si se define una tabla default, cuando queramos hacer consultas mediante los métodos entonces podremos evitarnos poner el nombre de la tabla como un argumento más.

INSTANCIANDO LA CLASE

Hacemos la instancia de la clase de la siguiente manera en php:

$ sqlCollection = new \sqllayer\SqlCollection($cfg);

Sin embargo si por algún motivo deseara pasar el array después entonces tendría que utilizar el siguiente método:

$ sqlCollection = new \sqllayer\SqlCollection();

$ sqlCollection ->setStr($cfg);

USO DE LOS MÉTODOS PARA EJECUTAR UNA SENTENCIA

CONDICIONES

Para colocar una condición es necesario ejecutar otro método antes de solicitar la ejecución de la sentencia, ya que es necesario ayudar al programa a ubicar los placeholders de nuestra condicional. El método a utilizar luce de la siguiente manera:

set_cond($arr, $var, $val);

Donde el primer argumento es booleano e indica si lo que recibirá es un arreglo (true) o una serie de strings (false) con los placeholders.

set_cond(false, “user”, “james”);

set_cond(true, array(“:user” => “james”));

PASO A PASO

Primero definimos un arreglo el cual contendrá los placeholders y sus valores:

$values = array(":name" => "$_POST[‘name’]", ":password" => "$_POST[‘password’]");

Luego definimos un string con nuestra condición sin la cláusula “WHERE”, es importante no colocar “WHERE”:

$condition = "name = :name AND password = :password";

Una vez definido las condiciones, se procede a colocarlas en el programa mediante el siguiente método:

$sqlCollection->set_cond(true, $values);

SELECT

Para realizar peticiones a la base de datos con la sentencia SELECT tiene que utilizar el método get_reg () el cual tiene como parámetro:

- **Las columnas a solicitar:** las cuales son una cadena de caracteres descritas mediante comas (,).
- **La condicional usada:** en dado caso de que no exista condicional simplemente se le asigna false o null.
- **La tabla donde se realizara la consulta:** Recordar que si anteriormente se ha asignado una tabla default, es posible omitir la tabla.

La manera de utilizar el método es como se presenta a continuación.

```
$STH = $sqlCollection->get_reg("cols", "conds", "table");
```

Ejemplos sin condiciones

Obtener todas las columnas de una tabla:

```
$STH = $sqlCollection->get_ reg("*",  false, "table_users");
```

Obtener todas las columnas de una tabla con definición default en el array de configuración:

```
$STH = $sqlCollection->get_ reg("*");
```

Obtener algunas columnas de una tabla default

```
$STH = $sqlCollection->get_ reg("user, name, password");
```

EJEMPLOS CON CONDICIONES

```
$sqlCollection->set_cond(true, array(“:user” => “james”, “:password” => “1234”));

$condition = "user = :user AND password = :password";
```

Obtener todas las columnas con condición

```
$STH = $sqlCollection->get_all_regs("*",  $condition, "table_users");
```

Obtener todas las columnas con condición usando una tabla default

```
$STH = $sqlCollection->get_all_regs("*",  $condition);
```

UPDATE

Para utilizar la sentencia UPDATE, necesitaremos hacer uso de dos métodos, el primero se encarga de ajustar los valores de las columnas y las columnas mismas como si fueran placeholders. Y el segundo es el encargado en sí de la ejecución de la consulta a la base de datos.

El primer método es set_col(), y recibe como argumento dos cadenas de caracteres las cuales contendrán las columnas y posteriormente los valores de dichas columnas.

```
$sqlCollection->set_col(”COLS”, ”VALUES”);

$STH =  $sqlCollection->update_reg(“cond”, "table_users");
```

UPDATE sin condiciones

```
$sqlCollection->set_col(”user, password, name”, ”patrick935, 1234, Patricksio”);

$STH =  $sqlCollection->update_reg(false, "table_users");
```

UPDATE con condiciones

```
$sqlCollection->set_cond(true, array(“:user” => “james”, “:password” => “1234”));

$condition = "user = :user AND password = :password";


$sqlCollection->set_col(”user, password, name”, ”patrick935, 1524, Patricksio”);

$STH =  $sqlCollection->update_reg($condition, "table_users");
```

"UPDATE table_users SET user = patrick935, password = 1524, name = Patricksio WHERE user = james AND password = 1234";

INSERT

Para realizar un insert es necesario utilizar dos métodos. Uno para el seteo de las variables y sus datos, esto para los placeholders.

```
$this->sqlCollection->set_col(“cols”, “values”);

$this->sqlCollection->set_reg(“condition”, “table”);
```

Ejemplo sin condiciones

```
$sqlCollection->set_col(“id, user, name”, “381, perk12, Perklyng”);
```

$sqlCollection->set_reg(false, “table_users”);

Ejemplo sin condiciones con table default

$sqlCollection->set_col(“id, user, name”, “381, perk12, Perklyng”);

$sqlCollection->set_reg(false);

DELETE

Para eliminar un registro se utiliza el método delete_reg() al igual que los demás métodos es posible especificar que columnas vaciar de una tabla dada o si se ignora esta última entonces tratara de actuar sobre alguna tabla default. Si no se especifica columna alguna y se coloca Null, False o algún similar entonces eliminara todo el registro

delete_reg(“cols”, “condition”, “table”);

Ejemplo, eliminar registro

$sqlCollection->set_cond(true, array(“:user” => “james”, “:password” => “1234”));

$condition = "user = :user AND password = :password";

$sqlCollection->delete_reg(Null, “$condition”, “table_user”);

Eliminar una conexión

Para eliminar la conexión a la BD, simplemente debemos invocar el método shutdown_Connection()

$sqlCollection-> shutdown_Connection();

Solicitar el handler de la conexión PDO.

Para ello se utiliza el método get_handler().

$handler = $sqlCollection-> get_handler();

FETCHING DATA EN SELECT

La manera en que SQLML solicita los datos al dbms no interfiere en la labor normal de PDO en PHP, esto quiere decir que al momento de cambiar el modo de extracción de los datos debe de ser de modo natural.

Leer: http://mx1.php.net/manual/en/pdostatement.setfetchmode.php

Ejemplo: solicitando todas las columnas de una tabla.

```
$STH = $sqlCollection->get_ reg("*",  false, "table_users");

$STH ->setFetchMode(\PDO::FETCH_ASSOC);

 $row  = $STH ->fetch();
```

Imprimiendo los datos de ciertas columnas

```
Print($row['user']);

Print($row['name']);

Print($row['age']);

Print_r($row);
```

ERRORES DE PETICIONES

Cuando ocurre un error es necesario eliminar la almohadilla que se encuentra al principio del método queryExecutor(), de esta manera se activaran los errores si es que ocurre alguno. Por otro lado es recomendable desactivar esa funcionalidad.

Por defecto con o sin la almohadilla el sistema retornara FALSE, en caso de que haya ocurrido un error o no se haya encontrado los datos solicitados.

Así entonces cuando usted haga algo similar a:

```
$STH = $sqlCollection->get_ reg("*",  false, "table_users");
```

Podrá usted comprobar si ocurrió algún imprevisto de la siguiente manera:

```
If(!$STH){

Echo “ocurrió un error”;

}
```

DDL

Con SQLML es posible realizar de forma sencilla queries con DDL, simplemente crea una instancia de SQLCOLLECTION omitiendo en el array de configuración el nombre de la base de datos. Posteriormente realiza la siguiente operación:

1) Obtenemos el handler de la conexión:

```
$handler = $this->sqlCollection->get_handler();
```

2) Utilizamos el método query, pasandole como argumento el query ddl:

```
$stmt = $handler->query("SHOW DATABASES");
```

3) Aplicamos el fetch mode que necesitemos:

```
$stmt->fetchAll(PDO::FETCH_ASSOC)
```