1 Curso POO PHP PHP Data Objects (PDO)

1.1 Sumario

1.2 PHP Data Objects (PDO)

Se vas programar unha aplicación que utilice como sistema xestor de bases de datos MySQL, a extensión MySQLi é unha boa opción. Ofrece acceso a todas as características do motor de base de datos, á vez que reduce os tempos de espera na execución de sentenzas.

Non obstante, se no futuro tes que cambiar o SXBD por outro distinto, terás que volver programar gran parte do código desta. Por iso, antes de comezar o desenvolvemento, é moi importante revisar as características específicas do proxecto. No caso de que exista a posibilidade, presente ou futura, de utilizar outro servidor como almacenamento, deberás adoptar unha capa de abstracción para o acceso aos datos. Existen varias alternativas como ODBC, pero sen dúbida a opción máis recomendable na actualidade é **PDO**.

O obxectivo é que se chegado o momento necesitas cambiar o servidor de base de datos, as modificacións que debas realizar no teu código sexan mínimas. Mesmo é posible desenvolver aplicacións preparadas para utilizar un almacenamento ou outro segundo se indique no momento da execución, pero este non é o obxectivo principal de PDO. PDO non abstrae de forma completa o sistema xestor que se utiliza. Por exemplo, non modifica as sentenzas SQL para adaptalas ás características específicas de cada servidor. Se isto fose necesario, habería que programar unha capa de abstracción completa.

A extensión PDO debe utilizar un driver ou controlador específico para o tipo de base de datos que se utilice. Podes consultar os controladores dispoñibles na túa instalación de PHP na información que proporciona a función phpinfo.

PDO baséase nas características de orientación a obxectos de PHP pero, ao contrario que a extensión MySQLi, non ofrece un interface de programación dual. Para acceder ás funcionalidades da extensión tes que empregar os obxectos que ofrece, cos seus métodos e propiedades. Non existen funcións alternativas.

1.2.1 Establecemento de conexións

Para establecer unha conexión cunha base de datos utilizando PDO, debes instanciar un obxecto da clase PDO pasándoo os seguintes parámetros (só o primeiro é obrigatorio):

- orixe de datos (DSN). É unha cadea de texto que indica que controlador se vai utilizar e a continuación, separadas polo carácter dous puntos, os parámetros específicos necesarios polo controlador, como por exemplo o nome ou dirección IP do servidor e o nome da base de datos.
- nome de usuario con permisos para establecer a conexión.
- contrasinal do usuario.
- opcións de conexión, almacenadas en forma de array.

Por exemplo, podemos establecer unha conexión coa base de datos 'platega' da seguinte forma:

```
$db = new PDO('mysql:host=localhost;dbname=platega', 'usuario', 'abc123.');
```

Se como no exemplo, se utiliza o controlador para MySQL, os parámetros específicos para utilizar na cadea DSN (separada unhas doutras polo carácter punto e coma) a continuación do prefixo "mysql:" son os seguintes:

- host. Nome ou dirección IP do servidor.
- port. Número de porto TCP no que escoita o servidor.
- dbname. Nome da base de datos.
- unix_socket. Socket de MySQL en sistemas Unix.

Se quixeses indicar ao servidor MySQL utilice codificación UTF-8 para os datos que se transmitan, podes usar unha opción específica da conexión:

```
$opcions = array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES utf8");
```

```
$db = new PDO('mysql:host=localhost;dbname=platega', 'usuario', 'abc123.', $opcions);
```

Unha vez establecida a conexión, podes utilizar o método **getAttribute** para obter información do estado da conexión e **setAttribute** para modificar algúns parámetros que afectan a esta. Por exemplo, para obter a versión do servidor podes facer:

```
$version = $db->getAttribute(PDO::ATTR_SERVER_VERSION);
print "Versión: $version";
```

E se queres por exemplo que che devolva todos os nomes de columnas en maiúsculas:

```
$db->setAttribute(PDO::ATTR_CASE, PDO::CASE_UPPER);
```

En PDO os erros notifícanse lanzando excepcións mediante obxectos da clase **PDOException** coa información necesaria para manexar a excepción.

```
try {
   $db = new PDO('mysql:host=localhost;dbname=platega', 'usuario', 'abc123.');
}
catch (PDOException $e) {
   print "Erro: " . $e->getMessage() . "<br/>";
   die();
}
```

Unha vez rematado o traballo coa base de datos, basta con eliminar as referencias ao obxecto para que se peche a conexión e se liberen os recursos.

```
$db = null;
```

1.2.2 Execución de consultas

Para executar unha consulta SQL utilizando PDO, debes diferenciar aquelas sentenzas SQL que non devolven como resultado un conxunto de datos, daquelas outras que si o devolven. No caso das consultas de acción, como INSERT, DELETE ou UPDATE, o método **exec** devolve o número de rexistros afectados.

Se a consulta xera un conxunto de datos, como é o caso de SELECT, debes utilizar o método **query**, que devolve un obxecto da **clase PDOStatement**.

```
$db = new PDO("mysql:host=localhost;dbname=platega", "usuario", "abc123.");
$resultado = $db->query("SELECT produto, unidades FROM stock");
```

1.2.3 Transaccións

Por defecto PDO traballa en modo "autocommit", isto é, confirma de forma automática cada sentenza que executa o servidor. Para traballar con transaccións, PDO incorpora tres métodos:

- **beginTransaction**. Deshabilita o modo "autocommit" e comeza unha nova transacción, que finalizará cando executes un dos dous métodos seguintes.
- **commit**. Confirma a transacción actual.
- **rollback**. Reverte os cambios levados a cabo na transacción actual.

Unha vez executado un commit ou un rollback, volverase ao modo de confirmación automática.

```
$ok = true;
$db->beginTransaction();
if($db->exec('DELETE ?') == 0) $ok = false;
if($db->exec('UPDATE ?') == 0) $ok = false;
?
if($ok) $db->commit();  // Se todo foi ben confirma os cambios
else $db->rollback();   //  e se non, revérteos
```

Se o motor do SXBD non soporta transaccións, como é o caso do MyISAM de MySQL, PDO executará o método beginTransaction sen erros, pero naturalmente non será capaz de reverter os cambios se fose necesario executar un rollback.

1.2.4 Obtención e utilización de conxuntos de resultados

Ao igual que coa extensión MySQLi, en PDO tes varias posibilidades para tratar co conxunto de resultados devolto polo método query. A máis utilizada é o método **fetch** da clase PDOStatement. Este método devolve un rexistro do conxunto de resultados, ou false se xa non quedan rexistros por percorrer.

```
$db = new PDO("mysql:host=localhost;dbname=platega", "usuario", "abc123.");
$resultado = $db->query("SELECT produto, unidades FROM stock");
while($rexistro = $resultado->fetch()) {
    echo "Produto ".$rexistro['produto']."": ".$rexistro['unidades']."<br />";
}
```

Por defecto, o método fetch xera e devolve a partir de cada rexistro un array con claves numéricas e asociativas. Para cambiar o seu comportamento, admite un parámetro opcional que pode tomar un dos seguintes valores:

- PDO::FETCH_ASSOC. Devolve só un array asociativo.
- PDO::FETCH_NUM. Devolve só un array con claves numéricas.
- PDO::FETCH_BOTH. Devolve un array con claves numéricas e asociativas. É o comportamento por defecto.
- PDO::FETCH_OBJ. Devolve un obxecto cuxas propiedades se corresponden cos campos do rexistro.

```
$db = new PDO("mysql:host=localhost;dbname=platega", "usuario", "abc123.");
$resultado = $db->query("SELECT produto, unidades FROM stock");
while ($rexistro = $resultado->fetch(PDO::FETCH_OBJ)) {
    echo "Produto ".$rexistro->produto."": ".$rexistro->unidades."<br />";
}
```

- PDO::FETCH_LAZY. Devolve tanto o obxecto coma o array con clave dual anterior.
- PDO::FETCH_BOUND. Devolve true e asigna os valores do rexistro a variables, segundo se indique co método **bindColumn**. Este método debe ser chamado unha vez por cada columna, indicando en cada chamada o número de columna (empezando en 1) e a variable a asignar.

```
$db = new PDO("mysql:host=localhost;dbname=platega", "usuario", "abc123.");
$resultado = $db->query("SELECT produto, unidades FROM stock");
$resultado->bindColumn(1, $produto);
$resultado->bindColumn(2, $unidades);
while ($rexistro = $resultado->fetch(PDO::FETCH_BOUND)) {
    echo "Produto ".$produto."": ".$unidades."<br />";
}
```

1.2.5 Consultas preparadas

Ao igual que con MySQLi, tamén utilizando PDO podemos preparar consultas parametrizar no servidor para executalas de forma repetida. O procedemento é similar e mesmo os métodos a executar teñen practicamente os mesmos nomes.

Para preparar a consulta no servidor MySQL, deberás utilizar o método **prepare** da clase PDO. Este método devolve un obxecto da clase **PDOStatement**. Os parámetros pódense marcar utilizando signos de consulta como no caso anterior.

```
$db = new PDO("mysql:host=localhost;dbname=platega", "usuario", "abc123.");
$consulta = $db->prepare('INSERT INTO cursos (cod, nome) VALUES (?, ?)');
```

Ou tamén utilizando parámetros con nome, precedéndoos polo símbolo de dous puntos.

```
$db = new PDO("mysql:host=localhost;dbname=platega", "usuario", "abc123.");
$consulta = $db->prepare('INSERT INTO cursos (cod, nome) VALUES (:cod, :nome)');
```

Antes de executar a consulta hai que asignar un valor aos parámetros utilizando o método **bindParam** da clase PDOStatement. Se utilizas signos de consulta para marcar os parámetros, o procedemento é equivalente ao método bindColumn de MySQLi.

```
$cod_produto = "G1301006";
$nome_produto = "POO para plataformas web con PHP e MySQL";
$consulta->bindParam(1, $cod_produto);
$consulta->bindParam(2, $nome_produto);
$consulta->execute();
```

Se utilizas parámetros con nome, debes indicar ese nome na chamada a bindParam.

```
$consulta->bindParam(":cod", $cod_produto);
$consulta->bindParam(":nombre", $nome_produto);
```

Unha vez preparada a consulta e enlazados os parámetros cos seus valores, execútase a consulta utilizando o método **execute**.

```
$consulta->execute();
```

Alternativamente, é posible asignar os valores dos parámetros no momento de executar a consulta, utilizando un array (asociativo ou con claves numéricas dependendo da forma en que indicases os parámetros) na chamada a execute.

```
$parametros = array(":cod" => "G1301006", ":nome" => "POO para plataformas web con PHP e MySQL");
$consulta->execute($parametros);
```

--Víctor Lourido 20:24 15 jul 2013 (CEST)