PDO – PHP Data Object

Advertência

Esta apostila é fruto de pesquisa por vários sites e autores e com alguma contribuição pessoal. Geralmente o devido crédito é concedido, exceto quando eu não mais encontro o site de origem. Isso indica que esta apostila foi criada, na prática, "a várias mãos" e não somente por mim. Caso identifique algo que seja de sua autoria e não tenha o devido crédito, poderá entrar em contato comigo para dar o crédito ou para remover: ribafs @ gmail.com (remova os dois espaços).

É importante ressaltar que esta apostila é distribuída gratuitamente no repositório:

https://github.com/ribafs/apostilas

Sugestões

Sugestões serão bem vindas, assim como aviso de erros no português ou no PDO. Crie um issue no repositório ou me envie um e-mail ribafs @ gmail.com.

Autor

Ribamar FS

ribafs @ gmail.com

https://ribamar.net.br

https://github.com/ribafs

Fortaleza, 16 de dezembro de 2021

Recomendações

O conhecimento teórico é importante para que entendamos como as coisas funcionam e sua origem, mas para consolidar um conhecimento e nos dar segurança precisamos de prática e muita prática.

Não vale muito a penas ser apenas mais um programador. É importante e útil aprender muito, muito mesmo, ao ponto de sentir forte segurança do que sabe e começar a transmitir isso para os demais.

Tenha seu ambiente de programação em seu desktop para testar com praticidade todo o código desejado.

Caso esteja iniciando com PDO recomendo que leia por inteiro e em sequência. Mas se já entende algo dê uma olhada no índice e vá aos assuntos que talvez ainda não domine.

Não esqueça de ver e testar também o conteúdo da pasta Anexos do repositório.

Caso tenha alguma dúvida, me parece que a forma mais prática de receber resposta é através de grupos. Depois temos o Google.

Dica: quando tiver uma dúvida não corra para pedir ajuda. Antes analise o problema, empenhe-se em entender o contexto e procure você mesmo resolver. Assim você está passando a responsabilidade para si mesmo, para seu cérebro, que passará a ser mais confiante e poderá te ajudar ainda mais. É muito importante que você confie em si mesmo, de que é capaz de solucionar os problemas. Isso vai te ajudar. Somente depois de tentar bastante e não conseguir, então procure ajuda.

Veja que este material não tem vídeos nem mesmo imagens. Isso em si nos nossos dias é algo que atrai pouca gente, pois vídeos e fotos são mais confortáveis de ler e acompanhar. Ler um texto como este requer mais motivação e empenho. Lembrando que para ser um bom programador precisamos ser daqueles capaz de se empenhar e suportar a leitura e escrita também.

Autodidata

Não tive a pretensão de que esta apostila fosse completa, contendo tudo sobre PDO, que seria praticamente impossível. Aquele programador que quando não sabe algo seja capaz de pesquisar, estudar, testar e resolver praticamente sozinho. Este é um profissional importante para as empresas e procurado.

Índice

1 – Introdução

Apareceu na versão 5.1 do PHP com uma extensão PECL. Como requer recursos de OO não deve rodar em versões anteriores do PHP.

A extensão PHP Data Objects (PDO) define uma interface leve e consistente para acessar bases de dados em PHP. Cada driver de base de dados que implementa a interface PDO pode expor características específicas das bases de dados como funções de extensão regulares. Note-se que não é possível executar qualquer função de base de dados usando a extensão PDO por si só; é necessário usar um driver PDO específico da base de dados para acessar a um servidor de bases de dados.

A PDO fornece uma camada de abstracção de acesso a dados, o que significa que, independentemente da base de dados que estiver a utilizar, utiliza as mesmas funções para emitir consultas e obter dados. A PDO não fornece uma abstracção de base de dados; não reescreve SQL ou emula características em falta. Deve utilizar uma camada de abstracção completa se precisar dessa facilidade.

A PDO acompanha o PHP.

https://www.php.net/manual/en/book.pdo.php

Traduzido com a versão gratuita do tradutor - www.DeepL.com/Translator

PDO (PHP Data Object) é um módulo nativo desenvolvido a partir do PHP 5 que tem como objetivo fornecer uma estrutura que facilita a integração de aplicações desenvolvidas sob o paradigma de orientação à objetos com bancos de dados relacionais de forma fácil e sem a necessidade de alterar a padronização do sistema.

Com o PDO é possível desenvolver aplicações mais seguras a nível de banco de dados. Dentre as funcionalidades que ele nos oferece podemos destacar o controle de transações e o uso de prepared statements.

Para ativar o módulo PDO no seu php basta ir ao arquivo de configuração php.ini, localizar as linhas php_pdo.dll e php_pdo_mysql.dll e descomentá-las (retirar o ponto e viírgula “;” do início da linha) . Nesse caso estaríamos aptos a utilizar PDO com MySQL. Se você utiliza outro SGBD, basta descomentar a linha referente à ele. Feito isso reinicie o seu servidor de aplicações.
https://www.rafaelwendel.com/2011/12/tutorial-pdo-php-data-object/

2 - Drivers de SGBDs suportados

O PDO é uma abstração de bancos de dados com suporte aos SGBDs:

- MySQL (PDO)
- PostgreSQL (PDO)
- SQLite (PDO)
- CUBRID (PDO)
- MS SQL Server (PDO)
- Firebird (PDO)
- IBM (PDO)
- Informix (PDO)
- MS SQL Server (PDO)
- Oracle (PDO)
- ODBC and DB2 (PDO)

https://www.php.net/manual/pt_BR/pdo.drivers.php

3 – Conexões

Exemplos de conexão

A mais simples

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

// ou

```
<?php
try {
   $dbh = new PDO('mysql:host=localhost;dbname=test', $user, $pass);
   print 'Conectou';
} catch (PDOException $e) {
   print "Error!: " . $e->getMessage() . "<br/>";
   die();
}
?>
```

Mais abrangente

```
<?php

class Connection
{
        private $host = 'localhost';
        private $db   = 'estoque';
        private $user = 'root';
        private $pass = 'root';
        public  $pdo;
        private $port = 3306;

   // Conexão com o banco de dados
   public function __construct(){
        try {
                $dsn = 'mysql:host='.$this->host.';dbname='.$this->db.';port='.$this->port;
                $this->pdo = new PDO($dsn, $this->user, $this->pass);
                // Boa exibição de erros
                $this->pdo->setAttribute(PDO::ATTR_EMULATE_PREPARES,false);
                $this->pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);

                $this->pdo->query('SET NAMES utf8');
                return $this->pdo;
        }catch(PDOException $e){
       // Usar estas linhas no catch apenas em ambiente de testes/desenvolvimento. Em produção apenas o exit()
                echo '<br><br><b>Código</b>: '.$e->getCode().'<hr><br>';
                echo '<b>Mensagem</b>: '. $e->getMessage().'<br>';
                echo '<b>Arquivo</b>: '.$e->getFile().'<br>';
                echo '<b>Linha</b>: '.$e->getLine().'<br>';
                exit();
        }
   }
}
```

4 - Constantes pré-definidas

Algumas das constantes pré-definidas

Constantes pré-definidas

As contantes abaixo são definidas por esta extensão e somente estarão disponíveis quando a extensão foi compilada com o PHP ou carregada dinamicamente durante a execução.

PDO::PARAM_BOOL (int)
  Represents a boolean data type.
PDO::PARAM_NULL (int)
  Represents the SQL NULL data type.
PDO::PARAM_INT (int)
  Represents the SQL INTEGER data type.
PDO::PARAM_STR (int)
  Represents the SQL CHAR, VARCHAR, or other string data type.
PDO::PARAM_STR_NATL (int)
  Flag to denote a string uses the national character set. Available since PHP 7.2.0
PDO::PARAM_STR_CHAR (int)
  Flag to denote a string uses the regular character set. Available since PHP 7.2.0
PDO::PARAM_LOB (int)
  Represents the SQL large object data type.

Mais detalhes:
https://www.php.net/manual/pt_BR/reserved.constants.php

5 – Atributos

Atributos do PDO

Para melhorar a segurança configure

```
$dbh->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
$dbh->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$options = [
    PDO::ATTR_ERRMODE => PDO::ERRMODE_EXCEPTION,
    PDO::ATTR_PERSISTENT => false,
    PDO::ATTR_EMULATE_PREPARES => true,
    PDO::ATTR_DEFAULT_FETCH_MODE => PDO::FETCH_ASSOC,
    PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES $charset COLLATE $collate"
];

try {
     $pdo = new PDO($dsn, $user, $pass, $options);
} catch (\PDOException $e) {
     throw new \PDOException($e->getMessage(), (int)$e->getCode());
}
```

Ou usando o método setAttribute:

```
<?php
$db = new PDO('mysql:host=localhost;dbname=testdb;charset=utf8mb4', 'username', 'password');
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$db->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);

PDO::FETCH_ASSOC
PDO::FETCH_BOTH (default)
PDO::FETCH_OBJ
```

https://www.php.net/manual/pt_BR/pdostatement.fetch.php

```
    $conn = new PDO($dbInfos, $username, $password);
    $conn->setAttribute(PDO::ATTR_EMULATE_PREPARES, false);
    $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
    $conn->setAttribute(PDO::ATTR_DEFAULT_FETCH_MODE, PDO::FETCH_ASSOC);
    $this->connection = $conn;
```

6 - Consultas sem preparedStatments

6.1 – query

Query - Prepara e executa uma instrução SQL sem placeholders/marcadores de lugar

```
$dbh = new PDO('mysql:host=localhost;dbname=testes', 'root', 'root');

$sql = 'SELECT nome, email FROM clientes ORDER BY nome';
foreach ($dbh->query($sql) as $row) {
    print $row['nome'] . " - ";
    print $row['email'] . "<hr>";
}
```

Outro

```
<?php
$sql = 'SELECT name, color, calories FROM fruit ORDER BY name';
foreach ($conn->query($sql) as $row) {
    print $row['name'] . "\t";
    print $row['color'] . "\t";
    print $row['calories'] . "\n";
}
?>
```

6.2 – exec

Exec - Executar uma instrução SQL e devolver o número de registros afetados

```
$dbh = new PDO('mysql:host=localhost;dbname=testes', 'root', 'root');

$count = $dbh->exec("DELETE FROM customers WHERE id = 10");

print("Excluído $count registro");
```

Ou

```
$dbh = new PDO('mysql:host=localhost;dbname=testes', 'root', 'root');

$count = $dbh->exec("DELETE FROM customers WHERE id > 10 and id < 15"); // Excluir do 11 ao 14

print("Excluídos $count registros");
```

Outro

```
<?php
$dbh = new PDO('odbc:sample', 'db2inst1', 'ibmdb2');

/* Delete all rows from the FRUIT table */
$count = $dbh->exec("DELETE FROM fruit");

/* Return number of rows that were deleted */
print("Deleted $count rows.\n");
?>
```

7 – Com PreparedStatment

Declarações preparadas

7.1 – BindValue

bindValue()

public PDOStatement::bindValue (mixed $parameter , mixed $value , int $data_type = PDO::PARAM_STR) : bool

```
<?php

$pdo = require 'connect.php';
$sql = 'insert into authors(first_name, last_name) values(?,?)';
$statement = $pdo->prepare($sql);

$statement->bindValue(':first_name', 'Nick');
$statement->bindValue(':last_name', 'Abadzis');

$statement->execute();


<?php

$pdo = require 'connect.php';

$sql = 'insert into authors(first_name, last_name) values(:first_name,:last_name)';
$statement = $pdo->prepare($sql);

$author = [
      'first_name' => 'Chris',
      'last_name' => 'Abani',
];

$statement->bindValue(':first_name', $author['first_name']);
$statement->bindValue(':last_name', $author['last_name']);

// change the author variable
$author['first_name'] = 'Tom';
$author['last_name'] = 'Abate';

// execute the query with value Chris Abani
$statement->execute();

$stmt = $pdo->prepare("SELECT full_name, gender FROM myTable WHERE id > ? LIMIT ?");
$stmt->bindValue(1, 39, PDO::PARAM_INT);
$stmt->bindValue(2, 23, PDO::PARAM_INT);
$arr = $stmt->fetchAll();
if(!$arr) exit('No rows');
var_export($arr);
$stmt = null;

$id = 25; //Final value
$stmt = $pdo->prepare("SELECT * FROM myTable WHERE id < ?");
$stmt->bindValue($id);
$id = 98;
$id = 39;
$stmt->execute();
```

```
$arr = $stmt->fetchAll();
if(!$arr) exit('No rows');
var_export($arr);
$stmt = null;
```

bindParam()

public PDOStatement::bindParam (mixed $parameter , mixed &$variable , int $data_type = PDO::PARAM_STR , int $length = ? , mixed $driver_options = ?) : bool

Exemplo:

```
<?php

$pdo = require 'connect.php';

$sql = 'insert into authors(first_name, last_name)
        values(:first_name,:last_name)';

$statement = $pdo->prepare($sql);

$author = [
        'first_name' => 'Chris',
        'last_name' => 'Abani',
];

$statement->bindParam(':first_name', $author['first_name']);
$statement->bindParam(':last_name', $author['last_name']);

// change the author variable
$author['first_name'] = 'Tom';
$author['last_name'] = 'Abate';

// execute the query with value Tom Abate
$statement->execute();
```

https://www.phptutorial.net/php-pdo/php-prepared-statement/

```
$id = 25;
$stmt = $pdo->prepare("SELECT * FROM myTable WHERE id < ?");
$stmt->bindParam($id);
$id = 98;
$id = 39; //Final value
$stmt->execute();
$arr = $stmt->fetchAll();
if(!$arr) exit('No rows');
var_export($arr);
$stmt = null;
```

Diferença entre bindValue e bindParam

No bindParam() o argumento esperado é uma referência (variável ou constante) e não pode ser um tipo primitivo como uma string ou número solto, retorno de função/método. Já bindValue() pode receber referências e valores como argumento.

Com bindParam:

```
$sex = 'male';
$s = $dbh->prepare('SELECT name FROM students WHERE sex = :sex');
$s->bindParam(':sex', $sex);
```

```
$sex = 'female';
$s->execute(); // Executado quando $sex = 'female'
```

Com bindValue:

```
$sex = 'male';
$s = $dbh->prepare('SELECT name FROM students WHERE sex = :sex');
$s->bindValue(':sex', $sex);
$sex = 'female';
$s->execute(); // Executado quando $sex = 'male'
```

No bindParam() o argumento esperado é uma referência(variável ou constante) não pode ser um tipo primitivo como uma string ou número solto, retorno de função/método. bindValue() pode receber referências e valores como argumento, basicamente é isso.

O bindParam sempre uso com variareis se usar uma valor tipo 'nome usuário' no lugar da variável vai dar erro pq ele não aceita valor, e o bindValue é o inverso eu uso o valor mesmo 'nome do usuário' ,1235456. Eu entendo assim bindParam parâmetros por referencia usa variável, bindValue valores mesmo direto sem uso de variável. Posso não estar certo em alguns pontos mas espero ajudar a clarear um pouco.
https://pt.stackoverflow.com/questions/87384/qual-a-diferen%C3%A7a-entre-bindparam-e-bindvalue

Passos para a criação de um PreparedStatments

Usando ? (interrogações)

```
$sql = 'insert into authors(first_name, last_name) values(?,?)';
$statement = $pdo->prepare($sql);
$statement->execute(['Sandra', 'Aamodt']);// Passar os valores para os ?
```

Usando named placeholders/lugares reservados

```
$sql = 'insert into authors(first_name, last_name) values(:first_name,:last_name)';
$statement->execute([
    'first_name' => 'Henry',
    'last_name' => 'Aaron'
]);
```

Opcionalmente:

```
$statement->execute([
    ':first_name' => 'Henry',
    ':last_name' => 'Aaron'
]);

$stmt = $pdo->prepare("SELECT * FROM users WHERE sex=?");
$stmt->execute([$sex]);
$data = $stmt->fetchAll();
```

ou encadeados

```
$data = $pdo→prepare($sql)→execute($params)→fetch();
```

7.2 – BindParam

PDOStatement::bindParam - Liga um parâmetro ao nome da variável especificada

Exemplos

```
<?php
/* Execute a prepared statement by binding PHP variables */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
   FROM fruit
   WHERE calories < :calories AND colour = :colour');
$sth->bindParam(':calories', $calories, PDO::PARAM_INT);
$sth->bindParam(':colour', $colour, PDO::PARAM_STR, 12);
$sth->execute();
?>
```

```
<?php
/* Execute a prepared statement by binding PHP variables */
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
   FROM fruit
   WHERE calories < ? AND colour = ?');
$sth->bindParam(1, $calories, PDO::PARAM_INT);
$sth->bindParam(2, $colour, PDO::PARAM_STR, 12);
$sth->execute();
?>
```

```
<?php
/* Call a stored procedure with an INOUT parameter */
$colour = 'red';
$sth = $dbh->prepare('CALL puree_fruit(?)');
$sth->bindParam(1, $colour, PDO::PARAM_STR|PDO::PARAM_INPUT_OUTPUT, 12);
$sth->execute();
print("After pureeing fruit, the colour is: $colour");
?>
```

7.3 - Usando arrays

/* Executar um prepared statement passando um array de valores sem os placeholders (':nome') */

/* Execute a prepared statement by passing an array of insert values */

```
$calories = 150;
$colour = 'red';
$sth = $dbh->prepare('SELECT name, colour, calories
   FROM fruit
   WHERE calories < ? AND colour = ?');
$sth->execute(array($calories, $colour));// Pra cá vem os parâmetros do where acima:calories e colour
```

/* Execute a prepared statement by passing an array of insert values */

```
$calories = 150;
$colour = 'red';
```

```
$sth = $dbh->prepare('SELECT name, colour, calories
  FROM fruit
  WHERE calories < :calories AND colour = :colour');
$sth->execute(array('calories' => $calories, 'colour' => $colour));


$anarray = array(42 => "foo", 101 => "bar");
$statement = $dbo->prepare("SELECT * FROM table WHERE col1 = ? AND col2 = ?");

//This will not work
$statement->execute($anarray);

//Do this to make it work
$statement->execute(array_values($anarray));


$data = array(
  array('name' => 'John', 'age' => '25'),
  array('name' => 'Wendy', 'age' => '32')
);

try {
  $pdo = new PDO('sqlite:myfile.sqlite');
}

catch(PDOException $e) {
  die('Unable to open database connection');
}

$insertStatement = $pdo->prepare('insert into mytable (name, age) values (:name, :age)');

// start transaction
$pdo->beginTransaction();

foreach($data as &$row) {
  $insertStatement->execute($row);
}

// end transaction
$pdo->commit();
```

//UPDATE OPERATION

```
$var1 = 'value1';
$var2 = 'value2';
$varint = 00;//integer value

$query = "UPDATE table_name SET col1=(:value1), col2=(:value2) WHERE col3=(:value3)";
$sql = $db->prepare($query);
$sql->execute(['value1'=>$var1, 'value2'=>$var2, 'value3'=>$var3]);
echo "UPDATED";
```

/* Executar um prepared statement passando um array de valores sem os placeholders (':nome') */

```
$sth = $dbh->prepare('SELECT name, colour, calories
   FROM fruit
   WHERE calories < ? AND colour = ?');
$sth->execute(array(150, 'red'));
$red = $sth->fetchAll();
$sth->execute(array(175, 'yellow'));
```

```
$yellow = $sth->fetchAll();
```

/* Executar um prepared statement passando um array de valores */

```
$sql = 'SELECT name, colour, calories
    FROM fruit
    WHERE calories < :calories AND colour = :colour';
$sth = $dbh->prepare($sql, array(PDO::ATTR_CURSOR => PDO::CURSOR_FWDONLY));
$sth->execute(array(':calories' => 150, ':colour' => 'red'));
$red = $sth->fetchAll();
$sth->execute(array(':calories' => 175, ':colour' => 'yellow'));
$yellow = $sth->fetchAll();
```

prepare statement inside loop:

```
for($i=0; $i<1000; $i++) {
    $rs = $pdo->prepare("SELECT `id` FROM `admins` WHERE `groupID` = :groupID AND `id` <> :id");
    $rs->execute([':groupID' => $group, ':id' => $id]);
}
```

prepare statement outside loop:

```
$rs = $pdo->prepare("SELECT `id` FROM `admins` WHERE `groupID` = :groupID AND `id` <> :id");
for($i=0; $i<1000; $i++) {
    $rs->execute([':groupID' => $group, ':id' => $id]);
}
```

8 - Exemplos de consultas

8.1 – Insert

```
$stmt = $pdo->prepare("INSERT INTO myTable (name, age) VALUES (?, ?)");
$stmt->execute([$_POST['name'], 29]);
$stmt = null;
```

Outro

```
$nome = $_POST['nome'];
$email = $_POST['email'];
$data_nasc = $_POST['data_nasc'];
$cpf = $_POST['cpf'];

  try{
     $stmte = $pdo->prepare("INSERT INTO clientes(nome,email,data_nasc,cpf) VALUES (?, ?, ?, ?)");
     $stmte->bindParam(1, $nome , PDO::PARAM_STR);
     $stmte->bindParam(2, $email , PDO::PARAM_STR);
     $stmte->bindParam(3, $data_nasc , PDO::PARAM_STR);
     $stmte->bindParam(4, $cpf , PDO::PARAM_STR);
     $executa = $stmte->execute();

     if($executa){
        echo 'Dados inseridos com sucesso';
                        header('location: index.php');
     }
     else{
        echo 'Erro ao inserir os dados';
     }
  }
  catch(PDOException $e){
    echo $e->getMessage();
  }
```

8.2 – Update

```
$stmt = $pdo->prepare("UPDATE myTable SET name = ? WHERE id = ?")->execute([$_POST['name'],
$_SESSION['id']]);
$stmt = null;
```

Outro

```
if(isset($_POST['enviar'])){
    $id = $_POST['id'];
    $nome = $_POST['nome'];
    $email = $_POST['email'];
    $data_nasc = $_POST['data_nasc'];
    $cpf = $_POST['cpf'];

    $sql = "UPDATE $tabela SET nome = :nome, email=:email, data_nasc=:data_nasc, cpf=:cpf WHERE id = :id";
    $sth = $pdo->prepare($sql);
    $sth->bindParam(':id', $_POST['id'], PDO::PARAM_INT);
    $sth->bindParam(':nome', $_POST['nome'], PDO::PARAM_INT);
    $sth->bindParam(':email', $_POST['email'], PDO::PARAM_INT);
    $sth->bindParam(':data_nasc', $_POST['data_nasc'], PDO::PARAM_INT);
```

```
    $sth->bindParam(':cpf', $_POST['cpf'], PDO::PARAM_INT);

   if($sth->execute()){
      print "<script>alert('Registro alterado com sucesso!');location='index.php';</script>";
    }else{
      print "Erro ao editar o registro!<br><br>";
    }
}
```

8.3 – Select

```
$stmt = $pdo->query("SELECT * FROM clients order by id desc");

while ($row = $stmt->fetch()) {

    $id = $row['id'];
    $name = $row['name'];
    $email = $row['email'];
    ?>
            print $id.'<br>';
            print $nome.'<br>';
            print $email.'<br>';
}
```

8.4 – Delete

```
$stmt = $pdo->prepare("DELETE FROM myTable WHERE id = ?");
$stmt->execute([$_SESSION['id']]);
$stmt = null;
```

Outro

```
$id = $_POST['id'];
    $sql = "DELETE FROM  $tabela WHERE id = :id";
    $sth = $pdo->prepare($sql);
    $sth->bindParam(':id', $id, PDO::PARAM_INT);
    if( $sth->execute()){
      print "<script>alert('Registro excluído com sucesso!');location='index.php';</script>";
    }else{
      print "Erro ao exclur o registro!<br><br>";
    }
```

9 – fetch

Fetch - Busca o registro (somente um) seguinte de um conjunto de resultados.

```
<?php
$pdo = new PDO('mysql:host=localhost;dbname=testes;charset=utf8mb4', 'root', 'root');

$sth = $pdo->prepare("SELECT name FROM customers");
$sth->execute();

print("PDO::FETCH_OBJ:<br> ");
$result = $sth->fetch(PDO::FETCH_OBJ);
print $result->name;

$sth = $dbh->prepare("SELECT name, colour FROM fruit");
$sth->execute();

/* Exercise PDOStatement::fetch styles */
print("PDO::FETCH_ASSOC: ");
print("Return next row as an array indexed by column name\n");
$result = $sth->fetch(PDO::FETCH_ASSOC);
print_r($result);
print("\n");

print("PDO::FETCH_BOTH: ");
print("Return next row as an array indexed by both column name and number\n");
$result = $sth->fetch(PDO::FETCH_BOTH);
print_r($result);
print("\n");

print("PDO::FETCH_LAZY: ");
print("Return next row as an anonymous object with column names as properties\n");
$result = $sth->fetch(PDO::FETCH_LAZY);
print_r($result);
print("\n");

print("PDO::FETCH_OBJ: ");
print("Return next row as an anonymous object with column names as properties\n");
$result = $sth->fetch(PDO::FETCH_OBJ);
print $result->name;
print("\n");
?>
```

10 – fetchAll

fetchAll - Devolve um array contendo todas os registros do conjunto de resultados.

```
$stmt = $pdo->prepare("SELECT * FROM myTable WHERE id <= ?");
$stmt->execute([5]);
$arr = $stmt->fetchAll(PDO::FETCH_ASSOC);
if(!$arr) exit('No rows');
var_export($arr);
$stmt = null;
```

ou

```
$arr = [];
$stmt = $pdo->prepare("SELECT * FROM myTable WHERE id <= ?");
$stmt->execute([5]);
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
  $arr[] = $row;
}
if(!$arr) exit('No rows');
var_export($arr);
$stmt = null;

$data = pdo($pdo, "SELECT * FROM users WHERE salary > ?", [$salary])->fetchAll();

$pdo = new PDO('mysql:host=localhost;dbname=testes;charset=utf8mb4', 'root', 'root');

$sth = $pdo->prepare("SELECT name FROM customers");
$sth->execute();

$result = $sth->fetchAll();
var_dump($result);
```

11 – rowCount e columnCount

rowCount – devolve a quantidade de registros de uma consulta

columnCount – devolve a quantidade de campos de uma consulta

rowCount

```
$dbh = new PDO('mysql:host=localhost;dbname=novo_banco', 'root', 'root');

function countAll($table){
    global $dbh;
   $sql = "select * from `$table`";

   $stmt = $dbh->prepare($sql);
    try { $stmt->execute();}
    catch(PDOException $e){echo $e->getMessage();}

return $stmt->rowCount();
}

print countAll('produtos');

$stmt = $pdo->prepare("UPDATE myTable SET name = ? WHERE id = ?");
$stmt->execute([$_POST['name'], $_SESSION['id']]);
echo $stmt->rowCount();
$stmt = null;
```

columnCount

```
$dbh = new PDO('mysql:host=localhost;dbname=testes', 'root', 'root');

$sth = $dbh->query("SELECT * FROM clientes");
//$sth = $dbh->query("SELECT nome, email FROM clientes");
$count = $sth->columnCount();

echo 'Campos '.$count;
```

Outro

```
   // LIMIT 1 Torna a consulta mais rápida, caso apenas retorne um registro
          public function numFields(){
      $sql = 'SELECT * FROM '.$this->table.' LIMIT 1';
                    $sth = $this->pdo->query($sql);
                    return $sth->columnCount();
          }
```

12 – errorCode

Códigos de erro do PDO e tratamento

Códigos de erro do MySQL
https://phpro.org/articles/MySQL-Error-Codes.html

Códigos de erro do PostgreSQL
https://www.postgresql.org/docs/12/errcodes-appendix.html

Códigos de erro do SQLite
https://www.sqlite.org/c3ref/c_abort.html

Entrada duplicada

```
try {
  $stmt = $pdo->prepare("INSERT INTO myTable (name, age) VALUES (?, ?)");
  $stmt->execute([$_POST['name'], 29]);
  $stmt = null;
} catch(Exception $e) {
  if($e->errorInfo[1] === 1062) echo 'Duplicate entry';
}
```

Código de erro

```
$dbh = new PDO('mysql:host=localhost;dbname=testes', 'root', 'root');// Erros aqui devem ser capturados com try/catch

$err = $dbh->prepare('SELECT skull FROM bones');// Tabela não existe
$err->execute();

echo "\nPDOStatement::errorCode(): ";
print $err→errorCode();
```

Outro

```
    try {
        $insertStr = $crud->insertStr();

        $sql = "INSERT INTO $table $insertStr";
        $sth = $crud->pdo->prepare($sql);

        for($x=1;$x<$numFields;$x++){
            $field = $crud->fieldName($x);
                        $sth->bindParam(":$field", $_POST["$field"], PDO::PARAM_INT);
                }
        $sth->execute();

         print "<script>location='./index.php?table=$table';</script>";
    } catch (Exception $e) {
        if($e->getCode() == '22007'){
            print '<h3>Data vazia ou inválida</h3>';
        }
                        echo '<br><br><b>Código</b>: '.$e->getCode().'<hr><br>';
                        echo '<b>Mensagem</b>: '. $e->getMessage().'<br>';// Usar estas linhas no catch apenas em ambiente de testes/desenvolvimento
                        echo '<b>Arquivo</b>: '.$e->getFile().'<br>';
                        echo '<b>Linha</b>: '.$e->getLine().'<br>';
```

```
                exit();
}
```

13 - LIKE e LIMIT

LIKE

Quando estamos selecionando valores no MySQL, em várias ocasiões não queremos um dado específico, mas valores que contenham parte do que gostaríamos de encontrar. LIKE pode receber dois caracteres curinga para substituir as partes dos valores que não sabemos ou queremos omitir. São eles:

- % – Que é equivalente qualquer valor independente da quantidade de caracteres;
- _ – Que é equivalente a apenas um caractere qualquer.

https://www.todoespacoonline.com/w/2014/10/utilizando-like-mysql/

Exemplos

```
$stmt = $pdo->prepare("SELECT id, name, age FROM myTable WHERE name LIKE %?%");

$search = "%{$_POST['search']}%";
$stmt = $pdo->prepare("SELECT id, name, age FROM myTable WHERE name LIKE ?");
$stmt->execute([$search]);
$arr = $stmt->fetchAll();
if(!$arr) exit('No rows');
var_export($arr);
$stmt = null;
```

Outro

```
if(isset($_GET['keyword'])){
   $keyword=$_GET['keyword'];

   $sql = "select * from clientes WHERE nome LIKE :keyword order by id";
   $sth = $pdo->prepare($sql);
   $sth->bindValue(":keyword", $keyword."%");
   $sth->execute();
         //$nr = $sth->rowCount();
   $rows =$sth->fetchAll(PDO::FETCH_ASSOC);
}
```

LIMIT

SQL: Limit
LIMIT é uma cláusula SQL que especifica o número de linhas que devem ser retornadas no resultado de uma consulta.

https://www.devmedia.com.br/sql-limit/41216

```
$sql = 'SELECT * FROM '.$this->table.' LIMIT 1';// LIMIT 1 - Torna a consulta mais rápida

if($sgbd == 'mysql'){
   $results = $pdo->prepare("SELECT * FROM clientes ORDER BY id LIMIT $start, $row_limit");
}else if($sgbd == 'pgsql'){
   $results = $pdo->prepare("SELECT * FROM clientes ORDER BY id LIMIT $row_limit OFFSET $start");
}
```

Muito utilizado em paginação de resultados, como acima.

14 – Transações

Uma transação é uma operação que executa tudo ou nada. Caso haja algum erro nada será executado.

Multiple Prepared Statements em Transações

```
try {
  $pdo->beginTransaction();
  $stmt1 = $pdo->prepare("INSERT INTO myTable (name, state) VALUES (?, ?)");
  $stmt2 = $pdo->prepare("UPDATE myTable SET age = ? WHERE id = ?");
  if(!$stmt1->execute(['Rick', 'NY'])) throw new Exception('Stmt 1 Failed');
  else if(!$stmt2->execute([27, 139])) throw new Exception('Stmt 2 Failed');
  $stmt1 = null;
  $stmt2 = null;
  $pdo->commit();
} catch(Exception $e) {
  $pdo->rollback();
  throw $e;
}
```

Outro

```
$insertStatement = $pdo->prepare('insert into mytable (name, age) values (:name, :age)');
// start transaction
$pdo->beginTransaction();
foreach($data as &$row) {
  $insertStatement->execute($row);
}

// end transaction
$pdo->commit();
try {
    $pdo->beginTransaction(); // Iniciar a transação
    $sql = "INSERT INTO customer ('"
    . implode("','", $fields) . "') VALUES (?,?,?,?,?,?)";
    $stmt = $pdo->prepare($sql);
    foreach ($data as $row) $stmt->execute($row);
    $pdo->commit(); // Executar tudo que estiver no block, entre o beginTransaction e esta linha
```

```
} catch (PDOException $e) {
    error_log($e->getMessage());
    $pdo->rollBack(); // Cancelar todas as operações do bloco acima
}
```

Outro exemplo:

```
try {
    $pdo->beginTransaction();
    $sql = "UPDATE conta_corrente SET saldo = saldo - 200 WHERE id = 15";
    $stmt = $pdo->prepare($sql);
    $stmt->execute();

    $sql2 = "UPDATE conta_corrente SET saldo = saldo + 200 WHERE id = 12";
    $stmt2 = $pdo->prepare($sql2);
    $stmt2 = $pdo->execute();

    $pdo->commit();
} catch (PDOException $e) {
    $pdo->rollback();
    error_log($e->getMessage());
}
```

Outro

```
try {
    $pdo->beginTransaction();
    $sql = "INSERT INTO customer ('"
    . implode("','", $fields) . "') VALUES (?,?,?,?,?,?)";
    $stmt = $pdo->prepare($sql);
   foreach ($data as $row) $stmt->execute($row);
    $pdo->commit();
} catch (PDOException $e) {
    error_log($e->getMessage());
    $pdo->rollBack();
}
```

```
<?php
  $db = new PDO("mysql:host=localhost;dbname=banco", "root", "");
  $db->beginTransaction();

  $db->exec("UPDATE pedidos SET compra = 5641");
```

```
$db->exec("UPDATE clientes SET compra = 5641 ");
$db->exec("INSERT INTO logística(compra) VALUES (5641)");
// Caso tudo tenha dado certo
$db->commit();

// Caso não deu certo
$db->rollback();
```

PDO::beginTransaction

Turns off auto-commit mode and begins a transaction. The transaction begins with PDO::beginTransaction and will end when PDO::commit or PDO::rollback is called.

Syntax:

bool PDO::beginTransaction (void)

Return Value:

Returns TRUE on success or FALSE on failure.

Example

```
<?php
try {
   $dbhost = 'localhost';
   $dbname='hr';
   $dbuser = 'root';
   $dbpass = '';
   $connec = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);
}catch (PDOException $e) {
   echo "Error : " . $e->getMessage() . "<br/>";
   die();
}

$connec->beginTransaction();
$result = $connec->exec("INSERT INTO user_details (userid, password, fname, lname, gender, dtob, country, user_rating, emailid) VALUES
('abcd123', '123@John', 'John', 'ray', 'M', '1992-06-11', 'USA', '130', 'John123@example-site.com')");
$connec->commit();
echo $result;
?>
```

PDO::commit

Commits a transaction, returning the database connection to auto-commit mode until the next call to PDO::beginTransaction() starts a new transaction.

Syntax:

bool PDO::commit (void)

Return Value:

Returns TRUE on success or FALSE on failure.

Example:

PDO::errorCode

PDO::errorCode retrieves the SQLSTATE (a two characters class value followed by a three characters subclass value) associated with the last operation on the database handle.

Syntax:

mixed PDO::errorCode();

Return Value:

Returns a five-char SQLSTATE as a string, or NULL if there was no operation on the statement handle.

Example:

```
<?php
try{$dbuser = 'postgres';
   $dbpass = 'abc123';
   $host = 'localhost';
   $dbname='postgres';
   $connec = new PDO("pgsql:host=$host;dbname=$dbname", $dbuser, $dbpass);
}
catch (PDOException $e)
{
   echo "Error : " . $e->getMessage() . "<br/>";
   die();
}

$query = "SELECT * FROM user_details where genderr='M'";
$connec->query($query);
echo $connec->errorCode();
?>
```

PDO::rollBack

Rolls back the current transaction, as initiated by PDO::beginTransaction(). A PDOException will be thrown if no transaction is active.

Syntax:

bool PDO::rollBack (void)

Return Value:

TRUE if the method call succeeded, FALSE otherwise.

Example:

```
<?php
try {
   $dbhost = 'localhost';
   $dbname='hr';
   $dbuser = 'root';
   $dbpass = '';
   $connec = new PDO("mysql:host=$dbhost;dbname=$dbname", $dbuser, $dbpass);
}catch (PDOException $e) {
   echo "Error : " . $e->getMessage() . "<br/>";
   die();
}

$connec->beginTransaction();
$sth = $connec->exec("DROP TABLE user_detail");
$connec->rollback();
?>
```

Outro

```
try {
  $stmt = $pdo->prepare("INSERT INTO myTable (name, age) VALUES (?, ?)");
  if(!$stmt->execute(['Justin', 18])) throw new Exception('Stmt Failed');
  $stmt = null;

  $stmt = $pdo->prepare("SELECT * FROM myTable WHERE power_level > ?");
  if(!$stmt->execute([9000])) throw new Exception('Stmt Failed');
  $arr = $stmt->fetchAll();
  $stmt = null;
```

```
  try {
    $pdo->beginTransaction();
    $stmt1 = $pdo->prepare("INSERT INTO myTable (name, state) VALUES (?, ?)");
    $stmt2 = $pdo->prepare("UPDATE myTable SET age = ? WHERE id = ?");
    if(!$stmt1->execute(['Rick', 'NY'])) throw new Exception('Stmt 1 Failed');
    else if(!$stmt2->execute([27, 139])) throw new Exception('Stmt 2 Failed');
    $stmt1 = null;
    $stmt2 = null;
    $pdo->commit();
  } catch(Exception $e) {
    $pdo->rollback();
    throw $e;
  }
} catch(Exception $e) {
  error_log($e);
  exit('Error message that user can understand for this page');
}
```

15 - Outros

Retornando os drivers disponíveis no servidor atual

```
<?php
$drivers = PDO::getAvailableDrivers();

foreach($drivers as $drv){
    print $drv.'<br>';
}
```

Metadados
Informações sobre informações

<?php

```
$dbh = new PDO('mysql:host=localhost;dbname=testes', 'root', 'root');

$select = $dbh->query('SELECT COUNT(*) FROM customers');
$meta = $select->getColumnMeta(0);
//var_dump($meta);
print 'native_type: '.$meta['native_type'].'<br>';
print 'pdo_type: '.$meta['pdo_type'].'<br>';
print 'flags0: '.$meta['flags'][0].'<br>';
print 'table: '.$meta['table'].'<br>';
print 'name: '.$meta['name'].'<br>';
print 'len: '.$meta['len'].'<br>';
print 'precision: '.$meta['precision'].'<br>';
```

Boas práticas
Use PDO::query() caso vá usar apenas uma única vez.
Use PDO::prepare () e PDOStatement::execute() quando você precisar processar a mesma declaração várias vezes, mas usando diferentes
valores.

• PDO::PARAM_INT: Integer (whole number)
• PDO::PARAM_LOB: Binary (such as an image, Word document, or PDF file)
• PDO::PARAM_STR: String (text)
• PDO::PARAM_BOOL: Boolean (true or false)
• PDO::PARAM_NULL: Null

https://www.php.net/manual/en/book.pdo.php
https://www.treinaweb.com.br/blog/o-que-e-pdo-no-php

Download desta apostila e de outras

- PHP
- MySQL
- Laravel
- PHPOO
- MVC

https://github.com/ribafs/apostilas
Também disponível na seção Arquivos de grupos do Facebook