



**INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA**
RIO GRANDE DO SUL
Câmpus Feliz

PDO

Prof. Moser Fagundes

Programação III
ADS

PHP Data Objects (PDO)

Nesta aula, vamos estudar o PHP Data Objects (**PDO**). o qual define uma *interface* mais *amigável* para acessar bancos de dados em PHP.

- Código orientado a objetos.
- Mais seguro (mantido e atualizado pela comunidade).
- Provém maneira consistente para lidar com erros através do lançamento de exceções.

Criando BD – Exemplos

```
CREATE DATABASE aula16;  
USE aula16;
```

```
CREATE TABLE Pessoa (  
    id INTEGER AUTO_INCREMENT PRIMARY KEY,  
    nome VARCHAR(20) NOT NULL,  
    sobrenome VARCHAR(20) NOT NULL );
```


```
INSERT INTO Pessoa (nome,sobrenome)  
VALUES ('Carlos','Garcia');  
INSERT INTO Pessoa (nome,sobrenome)  
VALUES ('Maria','Antunes');  
INSERT INTO Pessoa (nome,sobrenome)  
VALUES('Cassio','Silva');  
INSERT INTO Pessoa (nome,sobrenome)  
VALUES('Marcela','Moraes');
```

Conectando ao MySQL com PDO

```
<?php
// Parametros do construtor PDO:
// mysql: – driver do Banco de Dados a ser usado
// host=localhost – endereco do servidor onde esta o BD
// dbname=aula16 – nome do BD (ver slide anterior)
// charset=utf8 – charset usado na comunicacao
// 'root' – usuario do BD
// ' ' – senha do usuario do BD
```

```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
```

```
// Codigo do programa...
?>
```



String de opções que diz qual driver usar, qual BD será acessado e qual charset será usado.


Conectando ao MySQL com PDO

```
<?php
// Parametros do construtor PDO:
// mysql: – driver do Banco de Dados a ser usado
// host=localhost – endereco do servidor onde esta o BD
// dbname=aula16 – nome do BD (ver slide anterior)
// charset=utf8 – charset usado na comunicacao
// 'root' – usuario do BD
// '' – senha do usuario do BD

$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );

// Veja as configuracoes de erro no slide 14
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

//Codigo do programa...
?>
```



Nesta linha, configuramos o modo que os erros devem ser exibidos

exemplo_slide_4.php

Executando SELECT com PDO (1)

SELECT com PDO (1º modo)

```
<?php
```

```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$resultado = $db->query('SELECT * FROM Pessoa');

foreach($resultado as $linha) {
    echo $linha['nome'] . ' ' . $linha['sobrenome'] . "<br>";
}
```

```
?>
```

Usamos o foreach para percorrer o \$resultado

Neste exemplo, usamos o nome das colunas para acessar os dados

exemplo_slide_6.php

Executando SELECT com PDO (2)

SELECT com PDO (2º modo)


```
<?php
```

```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$resultado = $db->query('SELECT * FROM Pessoa');

foreach($resultado as $linha) {
    echo $linha[0].' '.$linha[1].' '.$linha[2]."<br>";
}
```

```
?>
```



Neste exemplo, usamos o número da coluna no lugar do nome

exemplo_slide_7.php

Opções de indexação do array resultado de um SELECT

Parâmetros para o método `fetchAll`

- **PDO::FETCH_ASSOC**

Retorna as linhas do BD como um array associativo indexado com nomes de colunas do BD.

- **PDO::FETCH_NUM**

Retorna as linhas do BD como um array indexado de modo numérico, onde 0 é o índice da primeira coluna, 1 é o índice da segunda coluna, etc...

- **PDO::FETCH_BOTH**

Duplica os dados, retornando as linhas tanto como um array associativo como um array numérico. Evite essa opção pois ela ocupa o dobro de memória.

Executando SELECT com PDO (3)

SELECT com PDO (3º modo)

<?php

```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$resultado = $db->query('SELECT * FROM Pessoa');

// $linhas = $resultado->fetchAll(PDO::FETCH_NUM);
// $linhas = $resultado->fetchAll(PDO::FETCH_BOTH);
$linhas = $resultado->fetchAll(PDO::FETCH_ASSOC);

print_r($linhas);
```

?>

Obtemos todas as linhas de uma única vez usando o método `fetchAll`

Por associação, índices são as strings correspondentes as colunas da tabela

Contando as linhas

Para contar as linhas com **PDO** usamos o método `rowCount()`

```
<?php
```


```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$resultado = $db->query('SELECT * FROM Pessoa');

$numeroLinhas = $resultado->rowCount();

echo $numeroLinhas.' linhas foram selecionadas!';
```

```
?>
```



Podemos contar as linhas retornadas por um SELECT ou afetadas por um INSERT, UPDATE ou DELETE.

INSERT – Método exec()

Exemplo do **INSERT** com PDO


```
<?php
```

```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$numeroLinhasInseridas = $db->exec(
    "INSERT INTO Pessoa (nome,sobrenome)
    VALUES ('Maria','Silva')");

echo $numeroLinhasInseridas." linha foi inserida.";
```

```
?>
```



O exec() retorna apenas o número de linhas afetadas, nada mais!

UPDATE – Método exec()

Exemplo do **UPDATE** com **PDO**


```
<?php
```

```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$numeroLinhasAtualizadas = $db->exec(
    "UPDATE Pessoa SET nome='Mike' WHERE sobrenome='Silva'");

echo $numeroLinhasAtualizadas." linhas foram atualizadas.";
```

```
?>
```



O exec() retorna apenas o número de linhas afetadas, nada mais

DELETE – Método exec()

Exemplo do **DELETE** com PDO


```
<?php
```

```
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$numeroLinhasApagadas = $db->exec(
    "DELETE FROM Pessoa WHERE sobrenome='Silva'");

echo $numeroLinhasApagadas." linhas foram apagadas.";
```

```
?>
```



O exec() retorna apenas o número de linhas afetadas, nada mais.

Lidando com erros usando PDO

O PDO tem 3 modos de lidar com erros:

PDO::ERRMODE_SILENT

Não exibe nenhuma mensagem de erro, você deve chamar o método `$db->errorInfo()` para obter informações sobre os erros ocorridos.

PDO::ERRMODE_WARNING

Apenas exibe warnings na tela, mas segue execução.

PDO::ERRMODE_EXCEPTION

Lança uma `PDOException` – é o modo que você vai querer usar pois permite que os erros sejam tratados.

Para determinar o modo usamos `$db->setAttribute()`

Lidando com erros usando PDO

Com **PDO** lidamos com erros usando a estrutura **try** e **catch**

```
<?php
    try {
        // Código que pode vir a gerar um erro
    } catch (PDOException $exception) {
        // Código para tratar erro caso aconteça
    }
?>
```

Lidando com erros usando PDO

Com **PDO** lidamos com erros usando a estrutura **try** e **catch**

```
<?php
$db = new PDO(
    'mysql:host=localhost;dbname=aula16;charset=utf8',
    'root', '' );
$db->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

try {
    $db->query( 'THUNDERCAT' );
} catch (PDOException $exception) {
    echo "<p><b> Ocorreu um erro! </b></p>";
    echo "<b>Mensagem</b>: ".$exception->getMessage();
}

echo "Programa segue normalmente depois do erro.";
?>
```

Essa query com erro lança uma exceção, a qual podemos tratar dentro do **catch** logo abaixo.