INSTITUTO FEDERAL
Sul-rio-grandense | Câmpus Pelotas

# PHP Data Objects (PDO)
# Aula 5

Professora Marla Cristina da Silva Sopeña

tsi tecnologia em sistemas para internet

# PHP Data Objects (PDO)

PDO veio para solucionar a migração de um banco de dados para outro, do MySQL para o PostgreSQL, por exemplo.

O PDO é uma camada de abstração de acesso a dados, onde os métodos de manipulação de dados são independentes do SGDB que você está utilizando, ou seja, podemos usar as mesmas funções para  qualquer banco.

O PDO veio no PHP 5.1 e dá suporte a vários sistemas gerenciadores de banco de dados, como MySQL, PostgreSQL, Oracle, SQL Server, IBM e etc.

A conexão com um banco de dados através do PDO se dá durante a criação de um objeto da classe PDO, passando informações de conexão com o banco na forma de um DSN (Data Source Name), além das credencias de acesso

# PHP Data Objects (PDO)

•É uma extensão que fornece uma interface padronizada para trabalhar com bancos de dados, cuja finalidade é prover um a API limpa e consistente, unificando a maioria das características presentes nas extensões de acesso a banco de dados.

•Unifica a chamada de métodos. Para conectar em banco de dados diferentes, a única mudança é na string de conexão:

Ex:

MySQL : new PDO(“mysql:host=localhost; dbname= bd; charset=utf8”, “usuario” , ”senha”)

Postgres: new PDO("pgsql:host=localhost; dbname= bd; user=usuario; password=senha")

# Exemplo conexão

```
try {

$pdo = new PDO("mysql:host=localhost; dbname=aula; charset=utf8", "root","");

$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

}

catch (PDOException  $e) {

   print $e->getMessage();

}
```

Instanciamento de um objeto da classe PDO - $pdo é o objeto criado que possibilita a conexão e execução de query no banco de dados

Método necessário para a exibição de erros do banco de dados

Através do try acontece uma tentativa de execução do código, caso aconteça um erro, este será tratado no bloco catch

# Vantagens de utilizar PDO

- Abstração de conexão e interação com banco de dados
- Segurança
- Suporte a diversos drivers

# Como usar o PDO?

Para utilizar o PDO, primeiro é instanciado um objeto da classe PDO, que representa a conexão com um banco.

Exemplo: $pdo = new PDO("pgsql:host=localhost; dbname= bd; user=usuario; password=senha");

Após abrir uma conexão, as consultas podem ser feitas de duas maneiras:

1) Através dos métodos "exec" ou o "query";

Exemplo: $pdo->exec(“insert into tabela values (‘$cod’)”);
Exemplo: $pdo->query(“select * from tabela”);

2) Montando uma prepared statement com o método "prepare", que devolve um objeto da classe PDOStatement, e depois utilizando o método "execute“ .

# Principais Métodos

Após abrir uma conexão podemos interagir com o banco utilizando 3 métodos da classe PDO:

| Método | Objetivo |
| --- | --- |
| exec | Utilizado para insert, update e delete.<br>Retorna  o número de linhas afetadas |
| query | Utilizado para resultados tabulares, comando select.<br>Devolve um objeto da classe PDO com o resultado |
| prepare | Utilizado para dados variáveis. |

# Como usar o PDO?

O método **"query"** é utilizado para consultas que retornam resultados tabulares (como o SELECT) e devolve um objeto da classe PDOStatement com o resultado.

O método **"exec"** é utilizado para consultas que não retornam resultados tabulares (como o INSERT, UPDATE, DELETE) e retorna apenas o número de linhas afetadas.

Estes métodos são úteis para executar consultas fixas (não-variáveis). Afinal, se envolvessem valores recebidos do usuário, estes valores precisariam ser validados (para evitar falhas de segurança com SQL Injection).

Já o método **"prepare"** é útil para montar uma consulta com dados variáveis. É possível especificar uma SQL com pontos de substituição que, ao serem substituídos, são escapados pela classe automaticamente.

# Como usar o PDO?

Prepared statements (método prepare) tendem a ser mais rápidas que as consultas convencionais, já que a consulta fica previamente "compilada" e pronta para execução com novos valores.

Ao invés do SGBD interpretar toda a SQL, ele apenas atribui novos valores aos pontos chave e realiza a operação. Funcionalidade muito útil para inserções ou atualizações em massa em uma tabela.

# Exemplo método prepare

```
<?php


$stmt=$pdo->prepare("INSERT INTO posts (titulo, conteudo) VALUES (?, ?)");

$stmt->execute(array("Arroz", "Meu primeiro item!"));
$stmt ->execute(array("Feijão", "Meu segundo item!"));
$stmt ->execute(array("Tomate", "Meu terceito item!"));

?>
```

Note que temos apenas uma query, mas iremos executar três vezes com três valores diferentes. Estamos passando um array de informações para o método execute, que pegará essas informações e colocará no lugar das interrogações.

# Existem diferentes formas de se executar uma prepared statement com PDO: (método prepare)

**1 - Usando "?" nos pontos-chave (de substituicao)**

```
$stmt = $pdo->prepare('INSERT INTO usuarios (nome, login) VALUES (?,?)');

// Passando os valores a serem usados no primeiro e segundo "?“
 $dados = array('Rubens da Silva', 'rubens');
 $inseriu = $stmt->execute($dados);
```

**2 - Usando pontos-chave nomeados e um array de dados**

```
$stmt = $pdo->prepare('INSERT INTO usuarios (nome, login) VALUES (:nome, :login)');

// Passando os valores a serem usados em :nome e :login
$dados = array(':nome' => 'Rubens da Silva ', ':login' => 'rubens');
$inseriu = $stmt->execute($dados);
```

# Existem diferentes formas de se executar uma prepared statement com PDO: (método prepare)

**3 - Usando pontos-chave nomeados e valores individuais**

```
 $stmt = $pdo->prepare('INSERT INTO usuarios (nome, login) VALUES (:nome, :login)');

// Utilizando o método bindValue para a atribuição de parametros.

$nome = 'Rubens da Silva';
$login='rubens';

$stmt->bindValue(':nome', $nome);
$stmt->bindValue(':login', $login);

// Executando a SQL com os valores definidos no bindValue

$inseriu = $stmt->execute();
```

# Usando exec

```
<?php

public function inserir(Categoria $categoria) {

 try {
        $inserir = $this->pdo->exec("INSERT INTO categoria (descricao) VALUES ('".$categoria->getDescricao."')");

        if ($inserir ==1)
        {
            echo "Categoria inserida com sucesso";
        }

    } catch (PDOException  $e) {
                print $e->getMessage(); }
    }
}

?>
```

# Usando query

```php
<?php

public function selecionar() {

 try {

        $stmt = $pdo->query('SELECT descricao FROM categoria');


        foreach ( $stmt  as $linha) // Percorrento um resultset
        {
            echo $linha[descricao];
        }


    } catch (PDOException  $e) {
                print $e->getMessage(); }

 }
?>
```

# Usando prepare para inserir novos registros (insert)

```php
<?php

public function inserir(Categoria $categoria) {

 try {
     $stmt = $pdo->prepare(
         'INSERT INTO categoria (codigo,descricao) VALUES (:cod,:desc)');

     $stmt->bindValue(':cod', $categoria->getCodigo());
     $stmt->bindValue(':desc', $categoria->getDescricao);

     $stmt->execute();

     } catch (PDOException  $e) {
                 print $e->getMessage(); }
 }
?>
```

# Usando prepare para selecionar registros (select)

```php
<?php
try {

$sql = "SELECT * FROM categoria WHERE cod_categoria = :cod";


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

   $res->bindValue(':cod', $categoria->getCodigo);
   $res->execute();

   echo $res['desc_categoria'];


} catch ( PDOException  $e) {

   print "Erro: Código:" . $e->getCode() . "Mensagem" . $e->getMessage(); }

?>
```

# Retornando dados com fetch e fetchAll

```php
<?php
public function selecionar_todos() {
$
$dados = $pdo->query("SELECT * FROM clientes");

$todos = $dados->fetchAll();

//retorna para a página todas as linhas da consulta (fetchAll)
return $todos;
}

public function selecionar_um($codigo) {

$dados = $pdo->query("SELECT * FROM clientes where codigo = '$codigo'");

$um = $dados->fetch();

//retorna apenas uma linha (fetch)
return $um;
?>
```

tsi tecnologia em sistemas para internet

# Percorrendo dados (fetchAll)

```
// percorre todas as linhas do resultado pelo foreach na página de chamada do método

$resultado= $ClienteDAO->selecionar_todos();

foreach ( $resultado as $linha)
    {
        echo "Nome: {$linha['nome']} - Usuário: {$linha['usuario']}<br />";
    }


?>
```

# Mostrando dados (fetch).

```
// mostra o resultado diretamente pelo vetor $resultado visto que o retorno é de apenas uma linha

$resultado= $ClienteDAO->selecionar_um();

echo $resultado['nome'];
echo $resultado['usuario'];


?>
```

# PDO::FETCH_ASSOC

```php
<?php
$
$dados = $pdo->query("SELECT * FROM clientes");

// percorre todas as linhas do resultado como um vetor associativo
while ( $linha = $dados->fetch( PDO::FETCH_ASSOC ) )
    {
        echo "Nome: {$linha['nome']} - Usuário: {$linha['usuario']}<br />";
    }


?>
```

# Retornos de resultados tabulares

**PDO::FETCH_ASSOC** – Retorna um array indexado pelo nome da coluna.

**PDO::FETCH_OBJ** – Retorna um objeto, de modo que cada coluna é acessada como uma propriedade.

Exemplo de retorno como objeto:

```
$result = $pdo->query(" SELECT * FROM clientes ");

if($result)
{
        //percorre os resultados via o fetch()
        while ($linha = $result->fetch(PDO::FETCH_OBJ))
        {

                echo $linha->nome . " - " . $linha->usuario . "<br>\n";

        }
}
```