PDO (PHP Data Objects)

por Rafael SP

O que é PDO (PHP Data Objects)?

PDO (**PHP** Data Objects) é um módulo de **PHP** montado sob o paradigma Orientado a Objetos e cujo objetivo é prover uma padronização da forma com que **PHP** se comunica com um banco de dados relacional. Este módulo surgiu a partir da versão 5 de **PHP.**

Com a chegada da quinta versão do PHP (na verdade na 5.1), nasceu uma poderosa ferramenta: o PHP Data Objects ou **PDO**. Uma biblioteca de drivers para conexão.

O PDO é uma camada de acesso a dados que como o **php.net** (orientado a objetos – OO) promete, "indiferente do banco de dados que esteja usando, você poderá usar as mesmas funções para executar queries ou pegar dados". O PDO consiste em unificar as interfaces de acesso a dados.

O que é PDO (PHP Data Objects)?

O PDO suporta nativamente as seguintes extensões: PostgreSQL, MySQL (3,4,5), Firebird, Sybase, Informix, Oracle, ODBC, DBLIB, IBM DB2, SQLite (2,3), MSSQL e FreeTDS.

A utilização do PDO fornece uma camada de abstração em relação a conexão com o banco de dados visto que o PDO efetua a conexão com diversos bancos de dados da mesma maneira, modificando apenas a sua string de conexão.

A classe PDO em sua instancia pede como parâmetro: o banco que será utilizado, o caminho do banco de dados e o nome da base de dados, após o login e a senha para acesso a este banco de dados. Exemplo:

O que é PDO (PHP Data Objects)?

```
BANCO DE DADOS:host=CAMINHO BANCO;dbname=NOME BASE
```

> Logo basta modificarmos essa string que teremos uma conexão com qualquer outro banco de dados.
>
> Para trabalhar com operações no banco como: incluir, excluir, alterar e pesquisar, o PDO possui um método conhecido como **prepare**. Como o próprio nome diz, este método apenas prepara uma operação no banco de dados, logo se faz necessário a utilização de outros métodos como o **execute** por exemplo, para realmente executar a operação.

Vantagens do PDO:

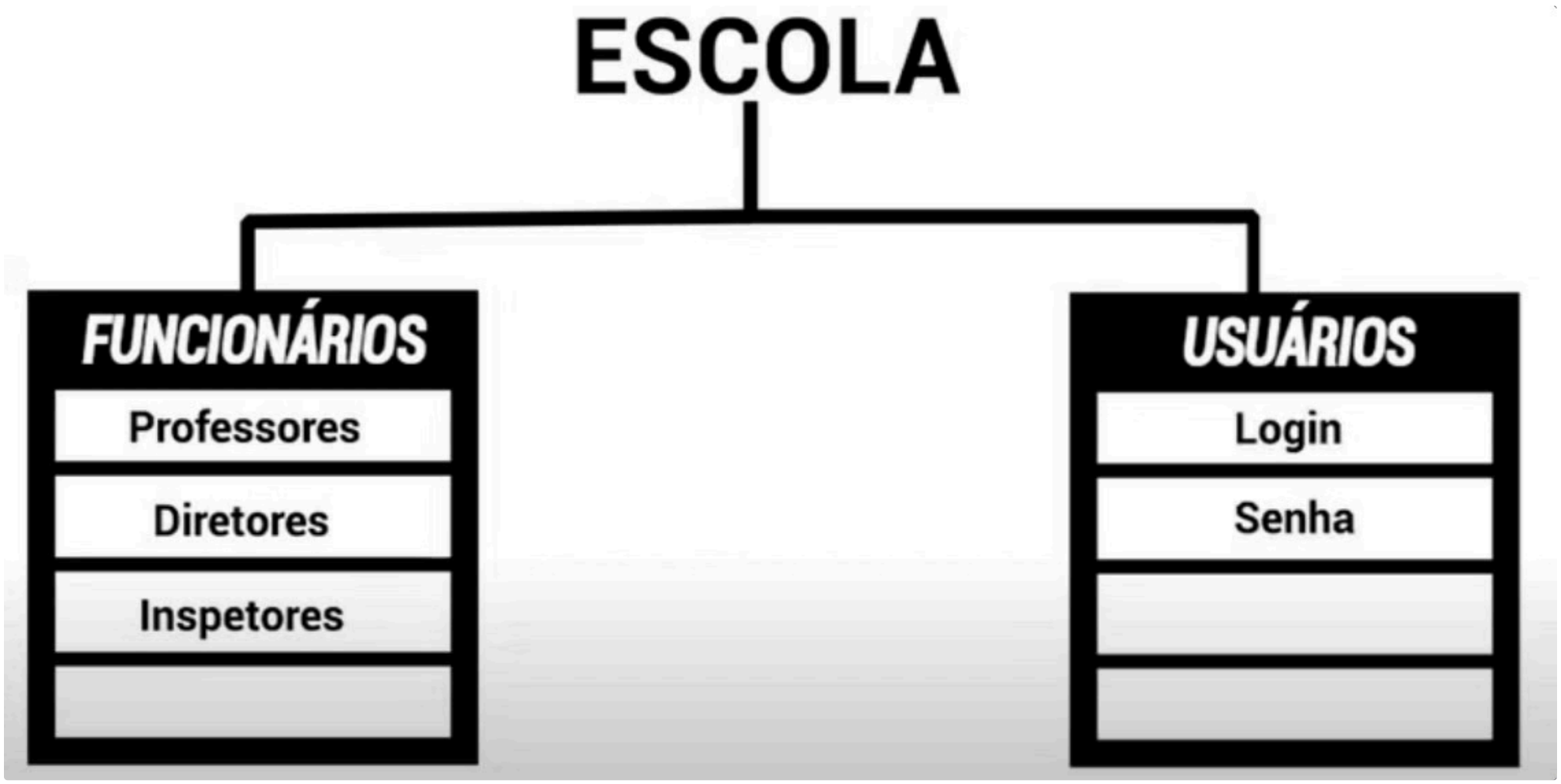
1. **Independência de Banco de Dados**: O PDO permite que você escreva código que funciona com múltiplos bancos de dados. Você pode trocar de banco de dados (por exemplo, de MySQL para PostgreSQL) sem alterar o código.
2. **Prepared Statements**: O PDO oferece suporte a prepared statements, o que significa que as consultas SQL podem ser preparadas e os parâmetros podem ser passados de maneira segura.
3. **Segurança**: Evita **SQL Injection**, uma das falhas mais comuns de segurança em aplicações web.
4. **Facilidade de Uso**: O PDO possui uma interface simples e poderosa.

SQL Injection

SQL Injection é uma técnica de ataque cibernético na qual um invasor insere código SQL malicioso em uma entrada de dados de um aplicativo da web, com o objetivo de manipular o banco de dados subjacente. Geralmente, os aplicativos da web usam consultas SQL para interagir com o banco de dados que armazena informações, como nomes de usuário, senhas, detalhes de transações etc.

Quando um aplicativo da web não valida ou **sanitiza** corretamente as entradas de usuário antes de usar esses dados em consultas SQL, ele se torna vulnerável a ataques de injeção SQL. O invasor pode inserir comandos SQL maliciosos em campos de entrada, como formulários de login ou campos de pesquisa, para manipular ou extrair dados do banco de dados, contornando as camadas de segurança.

Exemplo




```
SELECT * FROM usuarios WHERE login = ' "Rafa" ' AND senha ' "123456" '
```

```
$usuario = $_POST['usuario'];
$senha = $_POST['senha'];
$sql = "SELECT * FROM usuarios WHERE usuario = '".$usuario."' AND senha = '".$senha."' ";
$processa = mysql_query($sql);
```

```
SELECT * FROM usuarios WHERE login = ' "' OR ' 1 = 1" AND senha = ' "' OR ' 1 = 1"
```

' OR ' 1 = 1

Testes

Como usar PDO: Exemplos

Conectar ao Banco de Dados:

A primeira coisa que você faz com PDO é criar uma instância da classe PDO, que vai representar a conexão com o banco de dados.

```
<?php
try {
    // Conexão com o banco de dados
    $pdo = new PDO('mysql:host=localhost;dbname=nome_do_banco', 'usuario', 'senha');

    // Configurações adicionais de PDO
    $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION); // Ativa o modo de erro de exceção
    echo "Conexão bem-sucedida!";
} catch (PDOException $e) {
    // Caso ocorra erro na conexão
    echo "Erro na conexão: " . $e->getMessage();
}
?>
```

Executar uma Consulta Simples:

```
<?php
$query = 'SELECT * FROM usuarios';
$stmt = $pdo->query($query); // Executa a consulta

while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $row['nome'] . "<br>";
}
?>
```

Usando Prepared Statements (para evitar SQL Injection):

```
<?php
// Suponha que você tenha uma tabela 'usuarios' com uma coluna 'email'
$email = "exemplo@dominio.com";

// Prepara a consulta SQL com um parâmetro
$stmt = $pdo->prepare('SELECT * FROM usuarios WHERE email = :email');
$stmt->bindParam(':email', $email, PDO::PARAM_STR); // Ligando o parâmetro

$stmt->execute(); // Executa a consulta

// Exibe os resultados
while ($row = $stmt->fetch(PDO::FETCH_ASSOC)) {
    echo $row['nome'] . " - " . $row['email'] . "<br>";
}
?>
```

Inserir Dados no Banco de Dados com PDO:

```php
<?php
$nome = "João Silva";
$email = "joao.silva@exemplo.com";

// Prepara a consulta para inserir dados
$stmt = $pdo->prepare('INSERT INTO usuarios (nome, email) VALUES (:nome, :email)');
$stmt->bindParam(':nome', $nome);
$stmt->bindParam(':email', $email);

// Executa a consulta
if ($stmt->execute()) {
    echo "Usuário inserido com sucesso!";
} else {
    echo "Erro ao inserir usuário.";
}
?>
```

Links e Recursos para Aprofundar no Conceito:

1. **Documentação Oficial do PHP - PDO**:
   - **Documentação oficial do PDO** - A documentação oficial do PHP é sempre o melhor lugar para aprender detalhes e recursos da linguagem.
2. **Tutorial Completo de PDO**:
   - **Tutorial PDO - W3Schools** - Um tutorial simples e direto sobre como usar PDO com exemplos práticos.
3. **Cursos em Vídeo sobre PDO**:
   - **PDO e MySQLi no YouTube** - Um vídeo que explora a diferença entre PDO e MySQLi, além de cobrir o uso básico do PDO.
4. **Curso Completo de PHP com PDO**:
   - **Curso PHP e PDO - Udemy** - Curso pago, mas com conteúdo bem estruturado sobre como usar PDO em projetos PHP.
5. **Postagens e Discussões no StackOverflow**:
   - **PDO no StackOverflow** - Explore perguntas e respostas de desenvolvedores que enfrentaram problemas semelhantes.
6. **Blog sobre Programação PHP**:
   - **Como usar PDO no PHP (Blog Code Wall)** - Artigo bem detalhado com exemplos práticos de PDO.

Dicas para Melhor Uso do PDO:

1. **Sempre use Prepared Statements**: Eles são essenciais para evitar ataques de SQL Injection, além de tornarem o código mais eficiente.
2. **Gerencie Exceções**: Use `try-catch` para capturar exceções e fornecer mensagens de erro mais claras.
3. **Considere o Uso de Transações**: Se precisar realizar múltiplas operações no banco de dados, use transações para garantir que todas as operações sejam realizadas com sucesso, ou revertidas em caso de erro.
4. **Ajuste o Fetch Mode**: Defina o modo de obtenção de resultados (como `PDO::FETCH_ASSOC`, `PDO::FETCH_OBJ`, etc.) conforme suas necessidades.