DIO
CEZAR
GO

PDO no PHP

O QUE É?

- PDO (PHP Data Objects) é uma extensão que fornece uma interface padronizada para trabalhar com bancos de dados;
- abstrair conexão e interações com o banco de dados;
- independente do banco, os comandos serão os mesmos;
- PDO não é uma abstração do banco;
- PDO não reescreve SQL;

DRIVERS

- não é possível executar os comandos somente com o PDO, precisa-se de um driver específico para acessar determinado banco de dados;
- cada banco pode prover um driver para PDO, porém nem todos os recursos são suportados;
    - analisar então custo/benefício;

VANTAGENS

- abstração de conexão e interação com o banco de dados;
- segurança → a prova de SQL injection <3
- suporte a diversos drivers;

MANIPULANDO CONEXÕES

- deve ser criada em uma instância da classe PDO;
- seu construtor recebe informações em uma string que é um parâmetro obrigatório;
- também pode-se configurar a conexão com parâmetros adicionais no construtor;

COMPOSER.JSON

```
$conn = new PDO(
      'mysql:host=localhost;dbname=example-pdo', 'root', '123456',
      array(
            PDO::ATTR_PERSISTENT => true
      )
);
```

MÉTODOS

- depois de obter uma conexão com o banco, podemos interagir com 3 principais métodos:
  - `exec` → utilizado para insert, update e delete;
  - `query` → utilizado para resultados tabulares, comando select;
  - `prepare` → cria um *prepared statement*, utilizado para dados variáveis;

PREPARED STATEMETS

- os prepared statements oferecem dois ótimos benefícios:
  - a query só precisa ser preparada uma vez, mas pode ser executada várias vezes;
  - os parâmetros não precisam ser escapados, pois o driver cuida disso automaticamente;
- esses benefícios significam duas coisas, agilidade e segurança, confira a criação de um *prepared statement*;

PREPARED STATEMENTS

EXEMPLO PREPARED STATEMENTS

```php
<?php
      $stmt = $conn->prepare(
            'INSERT INTO posts (title, content) VALUES (:title, :content)'
      );

      $title = 'Título do post';
      $content = 'Conteúdo do post';

      // opcionalmente pode-se utilizar bindParam

      $stmt->bindValue(':title', $title);
      $stmt->bindValue(':content', $content);
      $stmt->execute();
?>
```

BINDPARAM VS BINDVALUE

- a grande diferença entre esses métodos é que o `bindParam()` recebe o valor do parâmetro por referência, sendo este realmente setado no momento em que o método `execute()` do *prepared statment* for chamado;
- o que pode gerar problemas em alguns casos, mas também pode facilitar em outros, sendo assim dê preferência ao `bindValue()` para os casos básicos;

RESGATANDO OS DADOS

- para resgatar os dados de um comando select temos algumas alternativas:
  - `fetch()` → retorna a próxima linha do resultado;
  - `fetchAll()` → retorna um array com todos os resultados;
  - `fetchObject()` → retorna a próxima linha do resultado como objeto;
  - `fetchColumn()` → retorna uma coluna da próxima linha do resultado.

RESGATANDO DADOS

EXEMPLO PARA IMPRIMIR DADOS DO BANCO

```
<?php
      $stmt = $conn->prepare("SELECT * FROM posts");
      while($row = $stmt->fetch()) {
            print_r($row);
      }
?>
```

TRATAMENTO DE ERROS

- a linguagem PHP oferece os mesmo comandos `try{}` e `catch{}` para o tratamento de erros e exceções;
- em PDO, pode-se utilizar estes comandos para tratamento de erros caso ocorram;

TRATAMENTO DE ERROS

- `PDO::ERRMODE_SILENT` → esse é o tipo padrão utilizado pelo PDO, basicamente o PDO seta internamente o código de um determinado erro, podendo ser resgatado através dos métodos `PDO::errorCode()` e `PDO::errorInfo()`;
- `PDO::ERRMODE_WARNING` → além de armazenar o código do erro, este tipo de manipulação de erro irá enviar uma mensagem `E_WARNING`, sendo este muito utilizado durante a depuração e/ou teste da aplicação.
- `PDO::ERRMODE_EXCEPTION` → além de armazenar o código de erro, este tipo de manipulação de erro irá lançar uma exceção `PDOException`, esta alternativa é recomendada, principalmente por deixar o código mais limpo e legível.

TRATAMENTO DE ERROS

EXEMPLO TRATAMENTO DE ERROS

```
<?php
      $dsn = 'mysql:host=localhost;dbname=example-pdo';
      $user = 'root';
      $password = '123456';
      try {
            $conn = new PDO($dsn, $user, $password);
            $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
      } catch (PDOException $e) {
            echo $e->getMessage();
      }
?>
```

MATERIAIS COMPLEMENTARES

- https://goo.gl/sR2vEN