

Treinamento Fundamentos de PHP

Turma 2023/01 – Aula 6/8



1

Semana 2



Aula 6 – Banco de dados

- Extensões no PHP
- Extensão PDO
- Conexão com Banco de Dados
- Executando SQL
- SQL Injection
- Query parametrizada
- Transações

2

Recapitulando...



Na última aula vimos sobre:

- Programação Orientada a Objetos
 - Encapsulamento
 - Herança
 - Polimorfismo
 - Abstração
 - Atributos e métodos estáticos
 - Namespaces
 - Autoload
- Atividade 4: Refatorar Hangman para POO

3

POO



Benefícios:

- Reutilização de código
- Modularidade
- Encapsulamento
- Flexibilidade e extensibilidade
- Gerenciamento de complexidade

4

Procedural x Orientado a Objeto



```

aula6 > paradigmas > funcoesRetangulo.php > ...
1 <?php
2
3 // Código Procedural
4
5 function calcularAreaRetangulo($largura, $altura) {
6     return $largura * $altura;
7 }
8
9 function calcularPerimetroRetangulo($largura, $altura) {
10    return 2 * ($largura + $altura);
11 }

```

```

aula6 > paradigmas > Retangulo.php > ...
1 <?php
2
3 namespace TreinamentoPHP\Exemplos\Paradigmas\OO;
4
5 // Código Orientado a Objetos
6
7 class Retangulo {
8     public function __construct(private $largura, private $altura) { }
9
10    public function calcularArea() {
11        return $this->largura * $this->altura;
12    }
13
14    public function calcularPerimetro() {
15        return 2 * ($this->largura + $this->altura);
16    }
17 }

```

```

aula6 > paradigmas > index.php > ...
1 <?php
2
3 $largura = 5;
4 $altura = 3;
5
6 // Procedural
7 require_once 'funcoesRetangulo.php';
8
9 $area = calcularAreaRetangulo($largura, $altura);
10 $perimetro = calcularPerimetroRetangulo($largura, $altura);
11 echo "Área do retângulo: " . $area . "<br>";
12 echo "Perímetro do retângulo: " . $perimetro;
13
14
15 // Orientado a Objeto
16 require_once 'Retangulo.php';
17 use TreinamentoPHP\Exemplos\Paradigmas\OO\Retangulo;
18
19 $retangulo = new Retangulo($largura, $altura);
20 $area = $retangulo->calcularArea();
21 $perimetro = $retangulo->calcularPerimetro();
22 echo "Área do retângulo: " . $area . "<br>";
23 echo "Perímetro do retângulo: " . $perimetro;
24

```

Copyright© 2023 Accenture All Rights Reserved

5

5

Abstração e Polimorfismo



Interface

- Além das classes abstratas também podemos alcançar a abstração e polimorfismos com Interfaces.
- As interfaces permitem que você especifique quais métodos uma classe deve implementar.
- As interfaces facilitam o uso de várias classes diferentes da mesma maneira. Quando uma ou mais classes usam a mesma interface, isso é chamado de "polimorfismo".

https://www.php.net/manual/pt_BR/language.oop5.interfaces.php

Copyright© 2023 Accenture All Rights Reserved

6

6

Interface



```
<?php

// Declara a interface 'iTemplate'
interface iTemplate
{
    public function setVariable($name, $var);
    public function getHtml($template);
}
```

```
// Isso NÃO funcionará
// Fatal error: Class BadTemplate contains 1 abstract methods
// and must therefore be declared abstract (iTemplate::getHtml)
class BadTemplate implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }
}
```

```
// Implementa a interface
// Isso funcionará
class Template implements iTemplate
{
    private $vars = array();

    public function setVariable($name, $var)
    {
        $this->vars[$name] = $var;
    }

    public function getHtml($template)
    {
        foreach($this->vars as $name => $value) {
            $template = str_replace('{ ' . $name . ' }', $value, $template);
        }

        return $template;
    }
}
```

Copyright© 2023 Accenture All Rights Reserved

7

7

Interfaces x Classes Abstratas



As interfaces são semelhantes às classes abstratas. A diferença entre interfaces e classes abstratas são:

- Interfaces não podem ter propriedades, enquanto classes abstratas podem
- Todos os métodos de interface devem ser públicos, enquanto os métodos de classe abstrata são públicos ou protegidos
- Todos os métodos em uma interface são abstratos, portanto não podem ser implementados no código e a palavra-chave abstract não é necessária
- Classes podem implementar uma interface enquanto herdam de outra classe ao mesmo tempo

Copyright© 2023 Accenture All Rights Reserved

8

8

Interfaces x Classes Abstratas



```

1  <?php
2
3  namespace TreinamentoPHP\Exemplos\Abstracao;
4
5  interface IAnimal {
6      public function fazerBarulho();
7  }
8
9  abstract class Animal {
10     public string $apelido;
11     public int $idade;
12
13     public function __construct(string $apelido = 'Lessy', int $idade = 0) {
14         $this->apelido = $apelido;
15         $this->idade = $idade;
16     }
17
18     abstract public function caminhar();
19
20     public function comer() {
21         return "$this->apelido está comendo...";
22     }
23 }

```

```

30 class Cachorro extends Animal implements IAnimal {
31     public function fazerBarulho() {
32         return 'Au, au';
33     }
34     public function caminhar() {
35         return 'Caminhando...';
36     }
37 }
38
39 class Gato implements IAnimal {
40     public function fazerBarulho() {
41         return 'Miau';
42     }
43 }
44
45 $cachorro = new Cachorro(idade: 2, apelido: 'Rex');
46 echo PHP_EOL . $cachorro->apelido;
47 echo PHP_EOL . $cachorro->fazerBarulho();
48 echo PHP_EOL . $cachorro->comer();
49
50 $gato = new Gato();
51 echo PHP_EOL . $gato->fazerBarulho();

```

Copyright© 2023 Accenture All Rights Reserved

9

9

Namespaces



Veio para resolver dois problemas diferentes:

- Eles permitem uma melhor organização agrupando classes que trabalham juntas para realizar uma tarefa
- Eles permitem que o mesmo nome seja usado para mais de uma classe

Copyright© 2023 Accenture All Rights Reserved

10

10

Autoload



- Recurso que permite carregar automaticamente as classes necessárias sob demanda, à medida que são utilizadas no código, sem a necessidade de incluir manualmente os arquivos das classes.
- Com o autoload, é possível definir uma função de autoload personalizada que é invocada sempre que uma classe é encontrada, mas ainda não está carregada. Essa função de autoload recebe o nome da classe como argumento e é responsável por incluir o arquivo da classe correspondente.

11



- É recomendado ter uma classe por arquivo no PHP (e em muitas outras linguagens de programação) por várias razões:
- Organização e legibilidade
- Reutilização e modularidade
- Autoloading
- Ferramentas de desenvolvimento

12

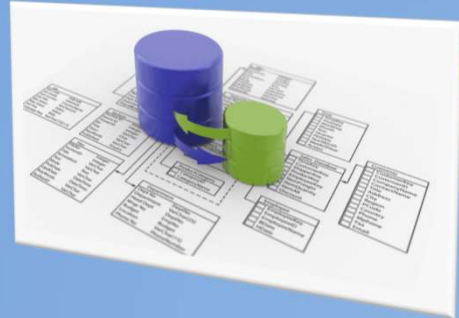
Banco de dados



Um banco de dados é uma coleção organizada de informações - ou dados - estruturadas, normalmente armazenadas eletronicamente em um sistema de computador.

Um banco de dados é geralmente controlado por um sistema de gerenciamento de banco de dados (DBMS).

[Oracle](#)

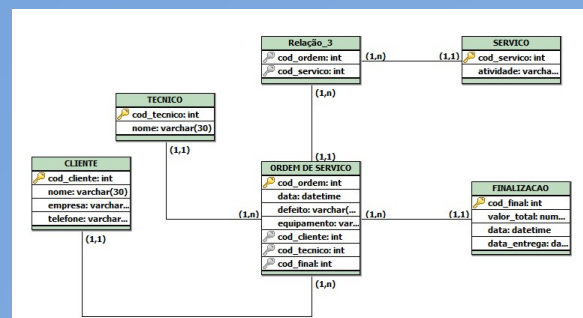
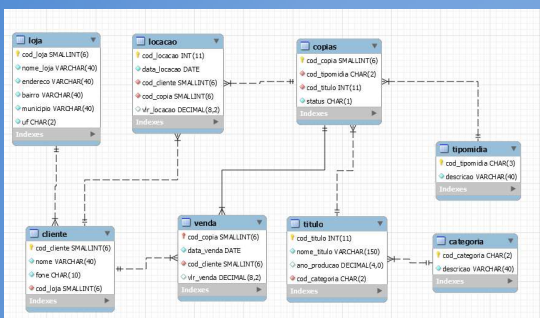


Copyright© 2023 Accenture All Rights Reserved

13

13

Banco de dados Relacional



Copyright© 2023 Accenture All Rights Reserved

14

14

Banco de dados



- Usaremos o SQLite neste curso
- Não há necessidade de nenhuma instalação
- SQLite é uma biblioteca de banco de dados embutida no PHP por padrão, o que significa que você não precisa instalá-lo separadamente para usá-lo com o PHP. Ele é um dos drivers de banco de dados suportados nativamente pelo PHP.
- A extensão sqlite3 oferece uma API rica para trabalhar com bancos de dados SQLite. Ela permite a criação, abertura, execução de consultas, manipulação de tabelas e outras operações relacionadas ao SQLite diretamente do código PHP.



Copyright© 2023 Accenture All Rights Reserved

15

15

Banco de dados



Extensões no PHP

- São módulos adicionais com funcionalidades extras à linguagem.
- São escritas em linguagem C que é compilado e vinculado ao PHP, permitindo estender a funcionalidade básica do PHP.
- As extensões podem ser fornecidas pela equipe do PHP ou por terceiros e são distribuídas como bibliotecas compartilhadas (.dll no Windows ou .so no Linux).
- Ao carregar uma extensão no PHP, você obtém acesso a novas funções, classes e recursos que não estão disponíveis no PHP básico.

Copyright© 2023 Accenture All Rights Reserved

16

16

Banco de dados



Extensão PDO (PHP Data Object)

- O PDO é uma extensão da linguagem PHP para acesso a banco de dados.
- Ele é totalmente orientado a objetos e possui diversos recursos importantes, além de suporte aos principais bancos de dados relacionais do mercado:
 - MySQL
 - SQL Server
 - PostgreSQL
 - Oracle
 - SQLite

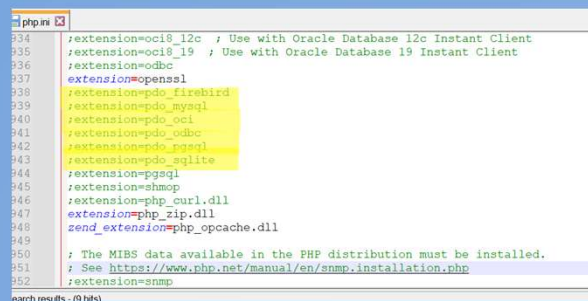
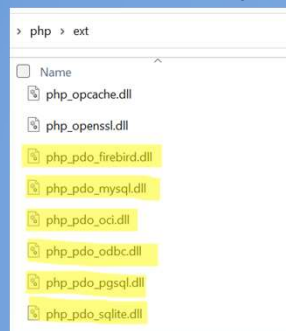
https://www.php.net/manual/pt_BR/pdo.drivers.php

17

Habilitando extensão



Para uma extensão funcionar em nosso ambiente, precisamos colocar o arquivo da extensão dentro da nossa máquina e indicar o caminho no php.ini.



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

18

Banco de dados



Criando conexão com o banco

```
<?php
    $conexão = new PDO($dsn, $username, $password, $options);
?>
```

Onde:

- string **\$dsn** (Data Source Name),
- string|null **\$username** = null,
- string|null **\$password** = null,
- array|null **\$options** = null

19

Banco de dados



Criando conexão com o banco

- Para conectar com o SQLite só precisamos passar o DSN nos parâmetros

```
1 <?php
2
3 $pdo = new PDO('sqlite:banco.sqlite');
4
```

https://www.php.net/manual/pt_BR/ref.pdo-sqlite.php

20

Banco de dados



Exemplos de conexão alguns bancos

```

1  <?php
2
3  $dsn = '';
4  $username = null;
5  $password = null;
6  $options = null;
7
8  // SQLite
9  $dsn = 'sqlite:treinamento.sqlite';
10 $conexao = new PDO($dsn, $username, $password, $options);
11
12 // MS SQLServer
13 $dsn = 'mssql:host=localhost;dbname=treinamento';
14 $username = 'root';
15 $password = 'P@ssw0rd';
16 $conexao = new PDO($dsn, $username, $password, $options);
17
18 // MySQL
19 $dsn = 'mysql:host=localhost:3307;dbname=treinamento';
20 $username = 'root';
21 $password = 'P@ssw0rd';
22 $options = [PDO::MYSQL_ATTR_INIT_COMMAND => 'SET NAMES utf8'];
23 $conexao = new PDO($dsn, $username, $password, $options);
24

```

Copyright© 2023 Accenture All Rights Reserved

21

21

Banco de dados



Principais funções:

- **PDO::exec()** - Executa uma instrução SQL e retornar o número de linhas afetadas
- **PDO::query()** - Prepara e executa uma instrução SQL sem parâmetros
- **PDO::prepare()** - Prepara uma instrução para execução e retorna um objeto de instrução
- **PDOStatement::execute()** - Executa uma instrução preparada

Copyright© 2023 Accenture All Rights Reserved

22

22

Para saber mais...



O **SQL** (Structured Query Language – Linguagem de Consulta Estruturada) possui comandos que permitem interagir com o banco de dados. São eles:

- Linguagem de Consulta de Dados
 - DQL (Data Query Language) possui apenas um único comando: **SELECT**.
- Linguagem de Definição de Dados
 - DDL (Data Definition Language) permite definir novas tabelas e elementos que serão associados a elas. Sendo composto por três comandos: **CREATE**, **ALTER** e **DROP**.
- Linguagem de Manipulação de Dados
 - DML (Data Manipulation Language) interage diretamente com os dados dentro das tabelas. Possui três comandos para esta manipulação: **INSERT**, **UPDATE** e **DELETE**.

Copyright© 2023 Accenture All Rights Reserved

23

23

Banco de dados



Criando tabela e inserindo dados

PDO::exec() - É indicado para instruções de definição e manipulação de dados (DDL e DML)

```

1  <?php
2
3  // Conectando com o SQLite
4  $conexao = new PDO('sqlite:treinamento.sqlite');
5
6  // Criando tabela users
7  $retorno = $conexao->exec('CREATE TABLE IF NOT EXISTS users (
8      id INTEGER PRIMARY KEY AUTOINCREMENT,
9      name TEXT,
10     email TEXT
11 ); ');
12 var_dump($retorno);
13
14 // Inserindo dados com EXEC
15 $sqlInsert = "INSERT INTO users (name, email) VALUES ('Fulano', 'fulano@emailemail.com')";
16 $retorno = $conexao->exec($sqlInsert);
17
18 var_dump($sqlInsert, $retorno);
19

```

Output da execução:

```

C:\curso-php-2023\inserrindo-dados-exec.php:12:1 int 0
C:\curso-php-2023\inserrindo-dados-exec.php:18:1 string 'INSERT INTO users (name, email) VALUES ('Fulano', 'fulano@emailemail.com')' (length=74)
C:\curso-php-2023\inserrindo-dados-exec.php:18:1 int 1

```

Copyright© 2023 Accenture All Rights Reserved

24

24

Banco de dados



Realizando consulta

PDO::query() - Prepara e executa uma instrução SQL em uma única chamada de função, retornando um objeto PDOStatement.

```
1 <?php
2
3 // Conectando com o SQLite
4 $conexao = new PDO('sqlite:treinamento.sqlite');
5
6 $sqlInsert = "SELECT * FROM users";
7
8 $retorno = $conexao->query($sqlInsert);
9 $retorno->setFetchMode(PDO::FETCH_ASSOC);
10
11 var_dump($retorno);
12
13 foreach ($retorno as $dado) {
14     var_dump($dado);
15 }
16
```

```
1 <?php
2
3 // Conectando com o SQLite
4 $conexao = new PDO('sqlite:treinamento.sqlite');
5
6 $sqlInsert = "SELECT * FROM users";
7
8 $retorno = $conexao->query($sqlInsert);
9 $dados = $retorno->fetchAll(PDO::FETCH_ASSOC);
10
11 var_dump($sqlInsert, $retorno, $dados);
12
```

<https://www.php.net/manual/en/class.pdostatement.php>

Copyright© 2023 Accenture All Rights Reserved

25

25

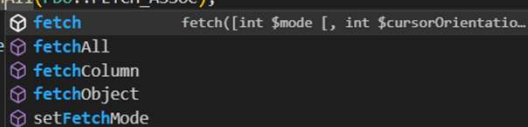
Banco de dados



O PDOStatement possui quatro métodos de busca

- fetch — Busca a próxima linha de um conjunto de resultados
- fetchAll — Busca as linhas restantes de um conjunto de resultados
- fetchColumn — Retorna uma única coluna da próxima linha de um conjunto de resultados
- fetchObject — Busca a próxima linha e a retorna como um objeto

```
$dados = $retorno->fetchAll(PDO::FETCH_ASSOC);
var_dump($sqlInsert, $re
```



Copyright© 2023 Accenture All Rights Reserved

26

26

Banco de dados



SQL Injection

- É um dos tipos mais comuns de ataques que existem.
- Segurança da informação é muito importante e PHP tem diversas medidas para protegermos nossos sistemas, mas além de muito importante, essa é uma área que exige muito estudo.

27

Banco de dados



Queries parametrizadas

```

1 <?php
2
3 // Conectando com o SQLite
4 $conexao = new PDO('sqlite:treinamento.sqlite');
5
6 $valor = '1; DROP TABLE users; --';
7 $valor = '1 OR 1=1';
8 $sql = "SELECT * FROM users WHERE id = :id";
9
10 $consulta = $conexao->prepare($sql);
11 $consulta->bindValue(':id', $valor, PDO::PARAM_INT);
12 $consulta->execute();
13 $consulta->setFetchMode(PDO::FETCH_ASSOC);
14
15 var_dump($consulta);
16
17 foreach ($consulta as $dado) {
18     var_dump($dado);
19 }

```

```

1 <?php
2
3 // Conectando com o SQLite
4 $conexao = new PDO('sqlite:treinamento.sqlite');
5
6 $valor = '1; DROP TABLE users; --';
7 $valor = '1 OR 1=1';
8 $sql = "SELECT * FROM users WHERE id = ?";
9
10 $consulta = $conexao->prepare($sql);
11 $consulta->bindValue(1, $valor, PDO::PARAM_INT);
12 $consulta->execute();
13 $consulta->setFetchMode(PDO::FETCH_ASSOC);
14
15 var_dump($consulta);
16
17 foreach ($consulta as $dado) {
18     var_dump($dado);
19 }

```

28

Banco de dados



Transações

- Transação é um conjunto de uma ou mais operações que compõem uma única tarefa ou unidade lógica de trabalho a ser executada.
- As transações são usadas para garantir a integridade dos dados em situações em que várias operações devem ser realizadas em conjunto e devem ter efeito completo ou nenhum efeito.
- Alguns exemplos de operações que podem ser realizadas em uma transação são transferências bancárias, reservas de ingressos ou atualizações de inventário

29

Banco de dados



Transações

```

1  <?php
2
3  try {
4      // Conectando com o SQLite
5      $conexao = new PDO('sqlite:treinamento.sqlite');
6      $conexao->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
7
8      $conexao->beginTransaction();
9
10     $conexao->query("INSERT INTO users (name, email) VALUES ('Angela', 'angela@email.com')");
11     $conexao->query("UPDATE users SET name = 'Ângela de Araujo' WHERE email='angela@email.com'");
12
13     $conexao->commit();
14
15     echo 'Usuário inserido com sucesso: ' . $conexao->lastInsertId();
16
17 } catch (\PDOException $e) {
18     $conexao->rollBack();
19     echo "Mensagem: {$e->getMessage()}";
20     echo "<br>Código: {$e->getCode()}";
21 }
22

```

30