



# Desenvolvimento Web II

## Linguagem PHP

*PHP Orientado a Objeto / Banco de Dados (PDO)*

[https://secure.php.net/manual/pt\\_BR/](https://secure.php.net/manual/pt_BR/)

**Gil Eduardo de Andrade**





# PHP Orientado a Objeto

## PHP Orientado a Objeto

- O PHP é uma linguagem de script que originalmente (quando criada) não possui suporte ao paradigma da programação orientado a objeto;
- O suporte a tal paradigma foi desenvolvido e introduzido a linguagem PHP a partir da sua versão 3.0;



# PHP Orientado a Objeto

## Classe e Construtor (codificação / interpretação / execução)

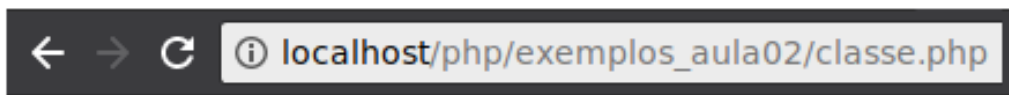
```
class aluno {  
    function __construct() {  
        echo "<h3> Classe: aluno -> Método: __construct() </h3>";  
    }  
}
```

Define uma nova classe com o nome "aluno".

Método construtor da classe, apresenta uma mensagem quando um objeto da classe é instanciado.

```
$obj = new aluno();
```

Instancia um objeto da classe "aluno".



Classe: aluno -> Método: \_\_construct()

Resultado da execução do código, método construtor é invocado quando o objeto da classe "aluno" é instanciado.

# PHP Orientado a Objeto

## Atributos (codificação)

```
class aluno {
```

```
    public $nome;
```

Define o atributo público "nome".

```
    function __construct($nome="VAZIO") {  
        $this->nome = $nome;  
    }  
}
```

Construtor recebe o nome do aluno como parâmetro, que por padrão armazena a string "VAZIO". O atributo nome recebe o parâmetro passado.

```
$obj = new aluno();  
echo "<h3>".$obj->nome."</h3>";
```

Objeto é instanciado sem passagem de parâmetro, atributo "nome" recebe a string "VAZIO".

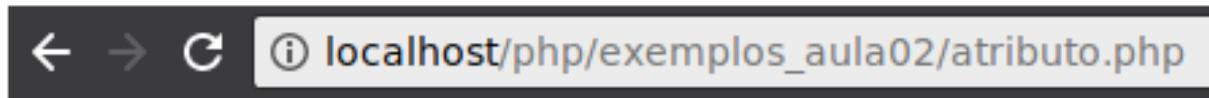
```
$obj = new aluno("MARIA");  
echo "<h3>".$obj->nome."</h3>";
```

Objeto é instanciado com passagem de parâmetro (MARIA), atributo "nome" recebe a string "MARIA".



# PHP Orientado a Objeto

## Atributos (interpretação / execução)



**VAZIO** →

Conteúdo do atributo “nome” para objeto instanciado sem passagem de parâmetro.

**MARIA** →

Conteúdo do atributo “nome” para objeto instanciado com passagem de parâmetro.



# PHP Orientado a Objeto

## Atributos e Métodos (codificação)

```
class aluno {
```

```
    private $nome; —————> Define o atributo privado "nome".
```

```
    function __construct($nome="Vazio") {  
        $this->nome = $nome;  
    }
```

```
    public function getNome() {  
        return $this->nome;  
    }
```

Define o método "*getNome()*" que permite acessar o atributo privado "*nome*" via um objeto da classe instanciado.

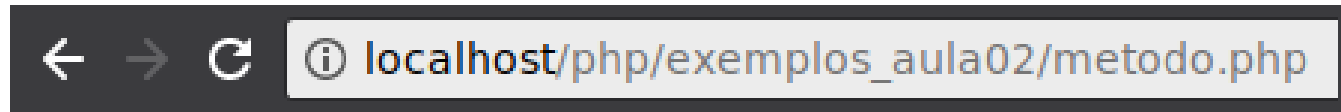
```
}  
  
$obj = new aluno();  
echo "<h3>".$obj->getNome()."</h3>";  
$obj = new aluno("Maria Eduarda Silva");  
echo "<h3>".$obj->getNome()."</h3>";
```

Como no exemplo anterior instancia dois objetos (com passagem de parâmetro e sem passagem de parâmetro). Contudo utiliza o método "*getNome()*" para acessar o atributo "*nome*".



# PHP Orientado a Objeto

## Atributos e Métodos (interpretação / execução)



**Vazio** →

Conteúdo do atributo “nome” (via método) para objeto instanciado sem passagem de parâmetro.

**Maria Eduarda Silva** →

Conteúdo do atributo “nome” (via método) para objeto instanciado com passagem de parâmetro.



# PHP Orientado a Objeto

## Atributos e Métodos (codificação)

```
class aluno {  
  
    private $nome;  
  
    function __construct($nome="VAZIO") {  
        $this->nome = $nome;  
    }  
  
    public function getReferencia() {  
        $partes = explode(" ", $this->nome);  
        return $this->getSobrenome().", ".$partes[0];  
    }  
  
    private function getSobrenome() {  
        $partes = explode(" ", $this->nome);  
        return mb_strtoupper($partes[count($partes)-1], 'UTF-8');  
    }  
}  
  
$obj = new aluno("Maria Eduarda Silva");  
echo "<h3>".$obj->getReferencia()."</h3>";
```

Define um método público *“getReferencia()”* capaz de invocar/utilizar o método privado *“getSobrenome()”* que não pode ser acessado externamente.

Define um método privado *“getSobrenome()”*.

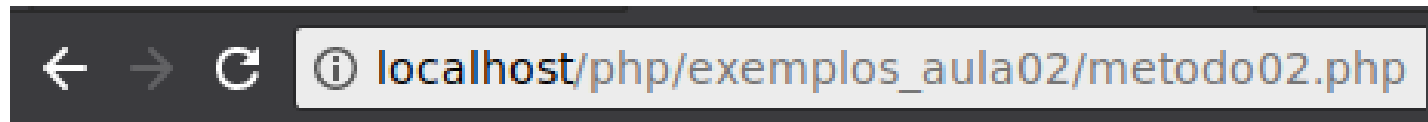
Invoca o público da classe.



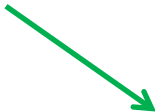


# PHP Orientado a Objeto

## Atributos e Métodos (interpretação / execução)



**SILVA, Maria**



Resultado da invocação do método público `getReferencia()` que faz uso do método privado `getSobrenome()` montar a *string* final retornada.



# PHP Orientado a Objeto

## Métodos Estáticos (codificação)

```
class aluno {  
  
    public static function getReferencia($nome) {  
        $partes = explode(" ", $nome);  
        return mb_strtoupper($partes[count($partes)-1], 'UTF-8').", ".$partes[0];  
    }  
}  
  
$ref = aluno::getReferencia("Maria Eduarda Silva");  
echo "<h3>".$ref."</h3>";
```

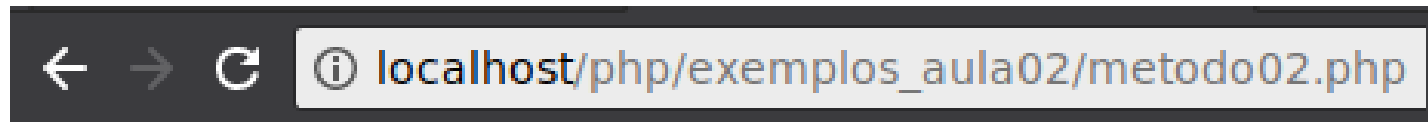
Define o método estático *“getReferencia()”* – a cláusula *“static”* permite que o método seja invocado sem que um objeto precise ser instanciado.

Invoca o método estático *“getReferencia()”* de maneira direta, sem instanciar um objeto da classe aluno, para tal utiliza o operador *“::”*. A utilização de métodos estáticos se dá quando o método não tem por objetivo modificar o estado interno de um objeto, ou seja, alterar o conteúdo de algum atributo. Observe que nesse caso a classe aluno nem possui atributos, ela possui apenas o método *“getReferencia()”* que recebe um nome como parâmetro, utilizado como dado de entrada para a rotina codificada dentro dele.

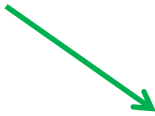


# PHP Orientado a Objeto

## Métodos Estáticos (interpretação / execução)



**SILVA, Maria**



Resultado da invocação do método estático *getReferencia()* sem que um objeto da classe aluno tenha sido instanciado para tal.



# PHP Orientado a Objeto

## Herança (codificação)

```
class pessoa {  
    public $nome;  
  
    function __construct($n) {  
        $this->nome = $n;  
    }  
    public function setNome($n) { $this->nome = $n; }  
    public function getNome() { return $this->nome; }  
}
```

Define a superclasse “*pessoa*” contendo o atributo “*nome*” e os métodos “*\_\_construct()*”, “*setNome()*” e “*getNome()*”.

```
class aluno extends pessoa {  
    public $turma;  
  
    function __construct($n, $t) {  
        parent::__construct($n);  
        $this->turma = $t;  
    }  
    public function setTurma($t) { $this->turma = $t; }  
    public function getTurma() { return $this->turma; }  
}
```

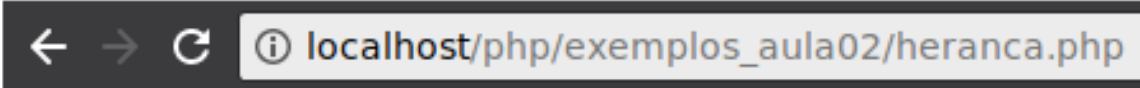
Define a subclasse “*aluno*” contendo o atributo “*turma*” e os métodos “*\_\_construct()*”, “*setTurma()*” e “*getTurma()*”. Ela herda os atributos e métodos da superclasse “*pessoa*”.

Observe que para invocar o construtor da superclasse é preciso utilizar a cláusula “*parent*”.

# PHP Orientado a Objeto

## Herança (interpretação / execução)

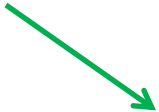
```
$obj = new aluno("Maria", "TADS16");  
echo "<h3>".$obj->getNome()."</h3>";  
echo "<h3>".$obj->getTurma()."</h3>";
```



← → ↻ ⓘ localhost/php/exemplos\_aula02/heranca.php

**Maria**

**TADS16**



Resultado da criação do objeto da subclasse *“aluno”* e chamada dos métodos *“getNome()”*, herdado da super classe *“pessoa”*, e *“getTurma()”* nativo da subclasse *“aluno”*.

# PHP Orientado a Objeto

## Herança: Estático (codificação)

```
class bd {  
    public static function select($tabela) {  
        return array("1" => "Maria",  
                    "2" => "Carlos",  
                    "3" => "João");  
    }  
}  
  
class modeloAluno extends bd {  
    public static function loadAlunos() {  
        return parent::select("tb_aluno");  
    }  
}
```

Define a superclasse “bd” contendo o método estático “select()” que recebe (na teoria) o nome da tabela do banco que será acessada para obter os dados.

Define a subclasse “modeloAluno” contendo o método estático “loadAlunos()”. Ela herda as funcionalidades da superclasse “bd”. Observe que para invocar o método “select()” da superclasse utilizamos a clausula “parent”.

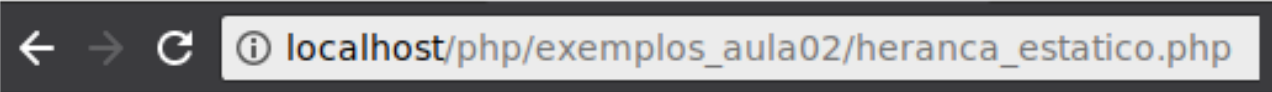


# PHP Orientado a Objeto

## Herança: Estático (interpretação / execução)

```
$dados = modeloAluno::loadAlunos();  
foreach($dados as $id => $nome) {  
    echo "<h3>".$id." - ".$nome."</h3>";  
}
```

---



1 - Maria

2 - Carlos →

3 - João

Resultado da invocação estática do método *loadAlunos()*, nativo da subclasse *modeloAluno*. O método *loadAlunos()* faz uso do método *select()*, herdado da super classe *bd*, que retorna o array (simula um *select* no banco) apresentado pelo comando *foreach*.





# Desenvolvimento Web II

**BANCO DE DADOS**  
**PHP DATA OBJECTS – PDO**







# Tipos de Dados - Constantes

## Introdução

- O PDO ou *PHP Data Objects* é uma extensão da linguagem PHP que fornece uma interface padronizada para interagir com banco de dados;
- O PDO tem como objetivo abstrair a interação entre a aplicação PHP e o banco, utilizando métodos genéricos que independem de qual banco de dados está sendo utilizado pela aplicação;



# PHP Orientado a Objeto

## Conexão / Select (codificação)

```
class BD {
```

```
private $DB_NOME = "aula02";  
private $DB_USUARIO = "root";  
private $DB_SENHA = "Gil.Eduardo12";  
private $DB_CHARSET = "utf8";
```

Define os atributos da classe que contém as informações necessárias para conectar com o banco de dados.

```
public function connection() {  
    $str_conn = "mysql:host=localhost;dbname=".$this->DB_NOME;  
  
    return new PDO($str_conn, $this->DB_USUARIO, $this->DB_SENHA,  
        array(PDO::MYSQL_ATTR_INIT_COMMAND => "SET NAMES ".$this->DB_CHARSET));  
}
```

String de conexão – MySQL

```
public function select() {  
    $conn = $this->connection();  
    $stmt = $conn->prepare("SELECT * FROM tb_alunos LIMIT 3");  
    $stmt->execute();  
  
    return $stmt;  
}
```

Instancia e retorna o objeto "PDO" que permite conectar com o banco e contém os métodos de interação com ele.

Método de seleção de dados do banco, efetua a conexão, prepara a "query" que será executada, efetua a sua execução e retorna um conjunto de objetos que contém os dados selecionados.

# PHP Orientado a Objeto

## Conexão / Select (interpretação / execução)

```
$obj = new BD();  
$alunos = $obj->select();  
  
while($objAluno = $alunos->fetchObject()) {  
    echo "<h4>".$objAluno->id." - ".$objAluno->nome."</h4>";  
}
```

← → ↻ ⓘ localhost/php/exemplos\_aula02/bd/conexao\_select.php

1 - MARIA EDUARDA DA SILVA

2 - DOUGLAS RODRIGO ALVES

11 - ANA PAULA MARQUES

Instancia um objeto da classe "BD", invoca o método "select()" que retorna um conjunto de objetos como resultado para seleção dos registros armazenados numa determinada tabela. A cláusula "fetchObject()" permite obter cada um dos registros no formato de um objeto.

# PHP Orientado a Objeto

## Insert (codificação)

```
public function insert($dados) {  
  
    $sql = "INSERT INTO tb_alunos(nome, curso, turma) VALUES(";  
  
    $flag = 0;  
    foreach($dados as $campo => $valor) {  
        if($flag == 0) {  
            $sql .= "'$valor'";  
            $flag = 1;  
        }  
        else { $sql .= ", '$valor'"; }  
    }  
    $sql .= ")";  
  
    $conn = $this->connection();  
    $stmt = $conn->prepare($sql);  
    $stmt->execute();  
  
    return $stmt;  
}
```

Monta a “query” de inserção no banco a partir do *array()* de dados recebido.

Efetua a conexão com o banco de dados, prepara a “query” para execução e efetua a inserção.

# PHP Orientado a Objeto

## Insert (interpretação / execução)

```
$dados = array("nome" => "MARCOS AURÉLIO",  
              "curso" => "INFORMÁTICA - EMI",  
              "turma" => "INFO14");
```

```
$obj = new BD();  
$aluno = $obj->insert($dados);  
echo "INSERIDO COM SUCESSO!";
```

Monta um *array()* com os dados que serão inseridos, instancia o objeto da classe "BD" e invoca o método "insert()", passando o *array()* de dados que foi criado.

← → ↻ ⓘ localhost/php/exemplos\_aula02/bd/insert.php

INSERIDO COM SUCESSO!



# Conceitos Iniciais

**Exemplos Utilizados no Documento** (Classe / Banco)

[http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii\\_exdoc02.zip](http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_exdoc02.zip)

**Exemplos Utilizados no Documento** (Aplicação Base Exercício)

[http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii\\_dica02.zip](http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_dica02.zip)

**Exercícios sobre o Conteúdo**

[http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii\\_pratica02.pdf](http://www.gileduardo.com.br/ifpr/dwii/downloads/dwii_pratica02.pdf)

