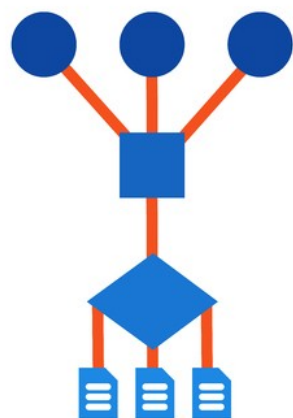


# Linguagem PHP

Prof. Daniel Di Domenico



Introdução e  
sintaxe dos comandos



# PHP

- Acrônimo de **PHP**: **H**ypertext **P**rocessor
  - Originalmente era **P**ersonal **H**ome **P**age
- Vastamente utilizada nos dias atuais para implementação de páginas e sistemas Web
  - É utilizada por grandes empresas, como Facebook, Wikipedia, WordPress
  - Estima-se que mais de 70% da internet roda em PHP
- Gratuita e livre
- Multiplataforma: uma aplicação PHP pode rodar em diversos sistemas operacionais (Windows, Linux e Mac)

- **Características da linguagem de programação PHP:**
  - Suporta programação estruturada e orientada a objetos
    - Faremos nossos projetos utilizando **orientação a objetos**
  - Possui alta integração com a linguagem HTML
    - É possível inserir códigos PHP dentro de uma página HTML, tornando-a dinâmica
  - É uma linguagem de *script* e interpretada
  - Linguagem *server-side* (roda em um servidor web)
    - JavaScript, por exemplo, é uma linguagem *client-side* (roda no lado cliente, ou seja, no navegador)
  - Facilidade de aprendizado

# PHP: requisitos de software

- **Pilha LAMP**

- Linux
- Apache
- MySQL
- PHP (versão 7 ou superior)



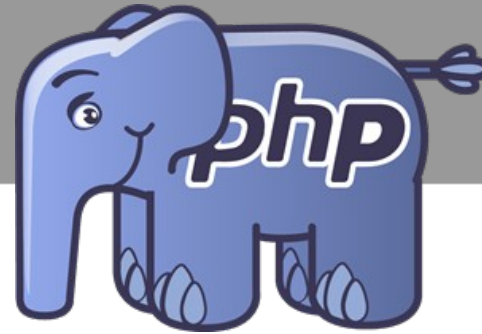
- **Visual Studio Code (VS Code):** editor de texto

- Instalar extensões:
  - PHP Intelephense (Ben Mewburn)
  - PHP Getters & Setters (Christophe VERGNE)
  - Path Intellisense (Christian Kohler)
- Desativar extensão:
  - PHP Language Features (digitar @builtin php)

# PHP: requisitos de software

- Apache + PHP + MySQL **para Windows**
  - **Xampp**  
([https://www.apachefriends.org/pt\\_br/index.html](https://www.apachefriends.org/pt_br/index.html))
  - **Wamp** (<https://www.wampserver.com/en/>)
  - **Atenção ao instalar:**
    - Verificar a versão do PHP (7 ou superior)
    - Armazenar o usuário e senha do MySQL
      - Você vai precisar destes dados depois
      - Cuidado se você já tiver outro serviço do MySQL rodando na sua máquina

# PHP: programação



- Tipagem dos dados:
  - Dinâmica: variáveis não precisam ser declaradas e assumem o tipo do valor atribuído
  - Fraca: pode-se alterar o tipo de uma variável
- PHP é **case-sensitive** (para praticamente tudo)
  - **Diferencia letras maiúsculas de minúsculas**
    - **Cuidado**: os programas ficarão com erro se o *case-sensitive* não for considerado
  - Boa prática: utilize o padrão camelCase

# PHP: programação

- Um programa PHP precisa ser escrito em um **arquivo .php** e dentro de um bloco:
  - Início do bloco: **<?php**
  - Fim do bloco: **?>**
    - O fim do bloco é opcional quando existe somente código PHP no arquivo

```
<?php
```

```
//Comandos PHP aqui
```

```
?>
```

# PHP: executando um programa

- Nossos programas PHP serão executados através do navegador
  - O código PHP será interpretado pelo servidor Apache
    - **O programa implementado deve estar na pasta do Apache**
- Como fazer um programa **Olá Mundo** em PHP?



```
Hello World.
```



# PHP: programação

- Exemplos de tipos de dados:

```
$var = "Aula em PHP"; //String
$var = 10; //Inteiro
$var = 3.1415; //Float
$var = true; //Booleano

//Identificar o tipo da variável
$tipo = gettype($var); //Retorna uma String com o tipo
```

- **Variáveis**
  - Identificadas pelo caractere \$ na frente do nome
- **Atribuição de valores à variáveis/objetos**
  - Caractere =
- **Final de instrução**
  - O PHP requer o uso de ; ao final de cada instrução
- **Strings**
  - Requer o uso " " ou ' ' de para identificar as Strings
  - Podem ser concatenadas com o operador . (ponto)

# PHP: programação

- Verificação de tipos de dados:

```
is_null($var); //Retorna true, pois a variável é nula/vazia

$var = "Aula em PHP";
is_string($var); //Retorna true, pois a variável é uma string

$var = 10; //Inteiro
is_integer($var); //Retorna true, pois a variável é um inteiro

$var = true; //Booleano
is_bool($var); //Retorna true, pois a variável é um booleano

//Demais: is_float(), is_array(), is_object(), is_numeric()
```

# PHP: comandos

- A linguagem PHP possui **comandos** para controlar o fluxo de execução do código, sendo:
  - Comando **IF** (se) para condições
    - Comando **ELSE** (senão) para condições compostas
  - Comando **SWITCH** (escolha) para condições com casos fixos com uma única entrada
  - Comando **WHILE** (enquanto) para repetições controladas por condições
  - Comando **FOR** (para) para repetições considerando um intervalo numérico pré-definido

# IF e ELSE (se e senão)

- Se a **condição** for verdadeira, o bloco é executado
  - ELSE é o caso contrário, sendo executado se as demais condições testadas nos IFs forem falsas

*//IF simples*

```
$n1 = 1;
```

```
$n2 = 2;
```

```
if($n1 == 1 && $n2 >= 2) {  
    echo "n1 = 1 e n2 >= 2";  
}
```

Condição

Em PHP, as **chaves** para abrir e fechar o bloco são opcionais caso exista apenas uma linha a ser executada

*//IF composto*

```
$n1 = 1;
```

```
if($n1 == 1) {  
    echo "n1 = 1";  
} else if($n1 == 2) {  
    echo "n1 = 2";  
} else {  
    echo "n1 != 1 e n1 != 2";  
}
```

Pode-se utilizar **else if** ou **elseif**

# PHP: programação

- Operadores relacionais:
  - == (igual)
  - === (idêntico – igual e de mesmo tipo)
  - !=, <> (diferente)
  - !== (não identico)
  - > (maior)
  - < (menor)
  - >= (maior ou igual)
  - <= (menor ou igual)
- Operadores lógicos
  - **and**, && (E)
  - **or**, || (OU)
  - **xor** (OU EXCLUSIVO)
  - **!** (NEGAÇÃO - not)

# SWITCH (escolha)

- Se a **variável** for igual ao **caso**, são executados os comandos do bloco
  - DEFAULT é o caso contrário. O uso do BREAK é obrigatório para finalizar o fluxo do comando SWITCH

Variável de  
teste para os  
casos

```
$n1 = 1;  
switch($n1) {  
    case 1:  
        echo "n1 = 1";  
        break;  
    case 2:  
        echo "n1 = 2";  
        break;  
    default:  
        echo "n1 != 1 e n1 != 2";  
}
```

# WHILE (enquanto)

- **Repete** a execução de um bloco de comandos enquanto a condição for **verdadeira**

```
$n1 = 1;
```

```
//while com condição avaliada no INÍCIO
```

```
while($n1 <= 5){  
    echo $n1 . "<br>";  
    $n1 = $n1 + 1;  
}
```

**Condição**

Executa **ao menos uma vez**,  
mesmo a condição sendo  
inicialmente FALSA

```
$n1 = 1;
```

```
//while com condição avaliada no FINAL
```

```
do {  
    echo $n1 . "<br>";  
    $n1 = $n1 + 1;  
} while($n1 <= 5);
```

**Condição**

Saída na tela:

1  
2  
3  
4  
5

# FOR (para)

- **Repete** a execução de um bloco de comandos enquanto a condição for **verdadeira**
  - Após cada iteração, ocorre a execução de uma expressão de incremento

**2- Condição de parada da repetição**

$n1++$  equivale a  
 $n1 = n1 + 1$

```
for( $n1=1; $n1<=5; $n1++ ) {  
    echo $n1 . "<br>";  
}
```

**1- Declaração da variável de controle da repetição**

Saída na tela:

1  
2  
3  
4  
5

**3- Incremento da variável de controle**



# Exercícios

- **1-** Faça um programa em PHP que imprima como saída os números de 1 a 10 de três formas:
  - Utilizando o comando WHILE
  - Utilizando o comando DO-WHILE
  - Utilizando o comando FOR
- **2-** Faça um programa que percorra os números de 1 a 100 e imprima a soma de todos eles como saída
- **3-** Faça um programa que percorra os números de 1 a 50 e imprima como saída apenas os números que sejam divisíveis por 3
  - Utilize o operador de resto de divisão: %
  - Exemplo:
    - $6 \% 3$  é igual a 0 – Então, 6 é divisível por 3
    - $7 \% 3$  é igual a 1 (1 é o resto da divisão) - Então, 7 NÃO é divisível por 3