Guia de PHP: Operadores, Estruturas Condicionais, Laços de Repetição e Superglobais

1. Operadores

Operadores Aritméticos

```
<?php

// OBJETIVO: Demonstrar como usar operadores matemáticos básicos em PHP

// CARACTERÍSTICAS: Realizam operações de soma, subtração, multiplicação, divisão e resto.


$a = 10;       // Cria a variável $a com valor 10

$b = 5;        // Cria a variável $b com valor 5


echo $a + $b;   // Soma: imprime 15

echo $a - $b;   // Subtração: imprime 5

echo $a * $b;   // Multiplicação: imprime 50

echo $a / $b;   // Divisão: imprime 2

echo $a % $b;   // Módulo (resto da divisão): imprime 0

?>
```

Operadores de Comparação

```
<?php

// OBJETIVO: Comparar valores e retornar verdadeiro (true) ou falso (false)

// CARACTERÍSTICAS: Usados para verificar igualdade, diferença e ordem (maior/menor).

$x = 10;       // Cria a variável $x com valor 10

$y = 20;       // Cria a variável $y com valor 20

var_dump($x == $y);  // Verifica se $x é igual a $y → false

var_dump($x != $y);  // Verifica se $x é diferente de $y → true

var_dump($x > $y);   // Verifica se $x é maior que $y → false

var_dump($x < $y);   // Verifica se $x é menor que $y → true
```

```
?>
```

Operadores Lógicos

```
<?php

// OBJETIVO: Verificar condições com operadores lógicos (E, OU, NÃO)

// CARACTERÍSTICAS: Permitem combinar condições booleanas.


$idade = 18;        // Cria a variável $idade com valor 18

$temCarteira = true;  // Cria a variável $temCarteira com valor verdadeiro


// Se idade for maior ou igual a 18 E a pessoa tiver carteira, entra no if

if ($idade >= 18 && $temCarteira) {

   echo "Pode dirigir!"; // Imprime essa mensagem

}

?>
```

2. Estruturas Condicionais

Definição: Controlam o fluxo do programa, escolhendo qual bloco de código executar dependendo de uma condição.

Características:

- Flexíveis para tomar decisões.
- Usadas para validar dados, autenticação, cálculos etc.

Vantagens:
Permite controlar o fluxo de execução.
Fácil leitura e entendimento.

Desvantagens:
Se usadas em excesso, deixam o código longo e confuso.
Muitos if/else podem ser substituídos por lógica mais simples.

If / ElseIf / Else

```php
<?php
// OBJETIVO: Classificar a nota em conceitos diferentes
$nota = 75;   // Cria a variável $nota com valor 75
if ($nota >= 90) {         // Se nota for maior ou igual a 90
   echo "Conceito A";      // Imprime "Conceito A"
} elseif ($nota >= 70) {    // Se não passou no if, mas nota >= 70
   echo "Conceito B";      // Imprime "Conceito B"
} else {
   echo "Conceito C";      // Caso contrário, imprime "Conceito C"
}
?>
```

Switch

```php
<?php
// OBJETIVO: Identificar a cor escolhida
$cor = "verde"; // Cria a variável $cor com valor "verde"
switch ($cor) {              // Verifica o valor de $cor
   case "vermelho":          // Se $cor for "vermelho"
      echo "A cor é vermelha";
      break;
   case "azul":              // Se $cor for "azul"
      echo "A cor é azul";
      break;
   default:                  // Se não for nenhum dos casos acima
      echo "Outra cor";
}
?>
```

3. Laços de Repetição

Definição: Executam um bloco de código várias vezes enquanto uma condição for verdadeira.

Características:

- Automatizam tarefas repetitivas.
- Podem percorrer listas e arrays.

Vantagens:
Reduzem repetição de código.
Tornam operações automáticas.

Desvantagens:
Podem gerar laços infinitos se a condição não for bem definida.
Excesso de loops aninhados pode comprometer desempenho.

While

```
<?php

// OBJETIVO: Contar de 1 até 5 usando while

// CARACTERÍSTICA: Executa enquanto a condição for verdadeira.

$i = 1;              // Cria a variável $i com valor 1

while ($i <= 5) {      // Enquanto $i for menor ou igual a 5

  echo "Número: $i <br>"; // Imprime o valor de $i

  $i++;              // Incrementa $i em +1

}

?>
```

For

```
<?php
// OBJETIVO: Contar de 1 até 5 usando for
// CARACTERÍSTICA: Ideal quando já se sabe o número de repetições.
for ($i = 1; $i <= 5; $i++) { // Início: 1, condição: <= 5, incremento: +1
   echo "Contagem: $i <br>"; // Imprime o valor de $i
}
?>
```

Foreach

```
<?php
// OBJETIVO: Percorrer um array e mostrar seus valores
// CARACTERÍSTICA: Criado especialmente para arrays.
$frutas = ["Maçã", "Banana", "Laranja"]; // Cria um array com frutas
foreach ($frutas as $fruta) {           // Para cada item no array
   echo "Fruta: $fruta <br>";         // Imprime o valor da fruta
}
?>
```

4. Variáveis Superglobais

$_GET

<!-- OBJETIVO: Enviar dados pela URL e recuperar com PHP -->

<!-- CARACTERÍSTICAS:

- Dados enviados pela URL (?chave=valor)

- Menos seguro, visível no navegador

- Bom para pesquisas e links rápidos

-->

<!-- arquivo form_get.html -->

<form action="processa_get.php" method="get"> <!-- Método GET envia os dados pela URL -->

Nome: <input type="text" name="nome"> <!-- Campo de texto chamado 'nome' -->

<input type="submit" value="Enviar"> <!-- Botão enviar -->

</form>

<!-- arquivo processa_get.php -->

<?php

// Aqui usamos $_GET → dados vêm da URL (ex: processa_get.php?nome=André)

echo "Olá, " . $_GET['nome'];

?>

$_POST

<!-- OBJETIVO: Enviar dados de forma mais segura (não aparece na URL) -->

<!-- CARACTERÍSTICAS:

- Dados não aparecem na URL

- Mais seguro que GET

- Usado para login, cadastro, formulários

-->

```
<!-- arquivo form_post.html -->

<form action="processa_post.php" method="post"> <!-- Método POST envia os dados escondidos -->

  Email: <input type="email" name="email"> <!-- Campo de email -->

  <input type="submit" value="Enviar">    <!-- Botão enviar -->

</form>

<!-- arquivo processa_post.php -->

<?php

// Aqui usamos $_POST → dados NÃO ficam visíveis na URL

echo "Email enviado: " . $_POST['email'];

?>
```

$_SERVER

```
<?php

// OBJETIVO: Mostrar informações do servidor e da requisição

// CARACTERÍSTICAS:

//   - Traz informações sobre o servidor e ambiente

//   - Mostra método usado (GET ou POST), IP, arquivo atual

//   - Útil para debug e monitoramento


echo "Arquivo atual: " . $_SERVER['PHP_SELF'] . "<br>";      // Nome do arquivo executado

echo "Nome do servidor: " . $_SERVER['SERVER_NAME'] . "<br>"; // Nome do servidor (ex: localhost)

echo "Método usado: " . $_SERVER['REQUEST_METHOD'] . "<br>";  // GET ou POST

?>
```

Diferenças principais

- **Estruturas Condicionais** → Controlam o fluxo (se / senão).
- **Laços de Repetição** → Repetem instruções várias vezes.
- **$_GET** → Dados pela URL, menos seguro.

- **$_POST** → Dados no corpo da requisição, mais seguro.
- **$_SERVER** → Informações do servidor e requisição.