



INSTITUTO FEDERAL
Sul de Minas Gerais

PHP – Introdução

Disciplina: Tecnologia Web

Prof. Me. Fernando Roberto Proença

O que é PHP?

2

- ❑ **PHP Hypertext Preprocessor** (ou simplesmente PHP) é uma linguagem de programação em *script open source*;
- ❑ Criada especialmente para desenvolvimento de sistemas Web;
- ❑ Executada do lado do servidor;
- ❑ Pode adicionar códigos HTML, CSS e JavaScript no PHP;
- ❑ Licença Gratuita;
- ❑ Multiplataforma;
- ❑ Suporte a vários Banco de Dados;
- ❑ Milhares de Páginas na internet foram criadas em PHP.

História do PHP

3

- ❑ Criador: **Rasmus Lerdorf**
- ❑ 1995 - PHP/FI
 - ▣ **Personal Home Page** / *Forms Interpreter*
 - ▣ Disponibilizou o código fonte para que qualquer um pudesse arrumar bugs e melhorar o código.
- ❑ 1997 – PHP/FI 2.0
 - ▣ Aproximadamente 50.000 domínios utilizando a linguagem, que representava 1% dos domínios da Internet.

História do PHP

4

- ❑ 1998 – PHP 3.0 - Andi Gutmans e Zeev Suraski
 - ▣ Foi rebatizado para PHP 3, acrônimo de Hypertext PreProcessor
 - ▣ Atingiu 10% dos domínios da Internet.
 - ▣ Não foi projetado para trabalhar com aplicações muito complexas eficientemente.
- ❑ 2000 – PHP 4
 - ▣ Utiliza novo mecanismo, conhecida como Zend Engine
 - ▣ Altíssimo ganho de performance

História do PHP

5

- 2004 – PHP 5
 - ▣ Utiliza o Zend Engine 2.0, adicionando ao core os conceitos da Orientação a Objetos.
 - ▣ Hoje com milhões de websites desenvolvidos que alcança 20% dos domínios da Internet no mundo.
- 2010 – PHP 6.0
 - ▣ Nem chegou a ser lançado oficialmente e foi cancelado em 2010.

História do PHP

6

- 2015 – PHP 7
 - ▣ Trouxe apenas melhorias em performance
 - ▣ Novas funcionalidades, além de implementar e fortificar novos recursos na orientação a objetos.
- 2019 – PHP 7.3.2 (versão mais atual)

Por que aprender PHP?

7

- ❑ Linguagem livre, gratuita e de fácil aprendizado
- ❑ Código-Fonte aberto
- ❑ Veloz, Robusta e Estável
- ❑ Portabilidade
 - Sistemas operacionais
 - Banco de dados
- ❑ Estruturada e Orientada a Objeto
- ❑ Fácil de aprender e fácil utilização
- ❑ Possuem muitos (e mais baratos) serviços de hospedagem

Por que aprender PHP?

8

- ❑ Recursos gratuitos
 - Servidor: Apache (Gratuito)
 - PHP: Atualmente PHP7 (Gratuito)
 - Banco de Dados: MySQL ou MariaDB (Gratuito)

Por que aprender PHP?

9

- Linguagens de programação mais populares do mundo em 2020:
 - ▣ **PHP – 4º lugar**
 - Fonte: <https://olhardigital.com.br/2020/03/03/pro/confira-as-20-linguagens-de-programacao-mais-populares-do-momento/>
- 12 linguagens de programação muito requisitadas no mercado brasileiro:
 - ▣ **PHP – 3º lugar – 13,67% das vagas**
 - Fonte: <http://exame.abril.com.br/carreira/12-linguagens-de-programacao-muito-requisitadas-no-mercado/>
- A quantidade de contribuições no GitHub em 2018:
 - ▣ **PHP – 5º lugar**
 - Fonte: <https://devsamurai.com.br/qual-linguagem-de-programacao-aprender-em-2018/>

A Comunidade PHP

10

- A comunidade oferece ajuda:
- Fóruns
- E-mails
- *“PHP é fácil, apenas um pouco mais complicado do que o HTML, mas provavelmente mais simples do que JavaScript ou ASP e definitivamente seu conceito é menos complexo do que JSP.” (PHP – A Bíblia)*
- Manual do PHP
 - ▣ <http://www.php.net/manual>

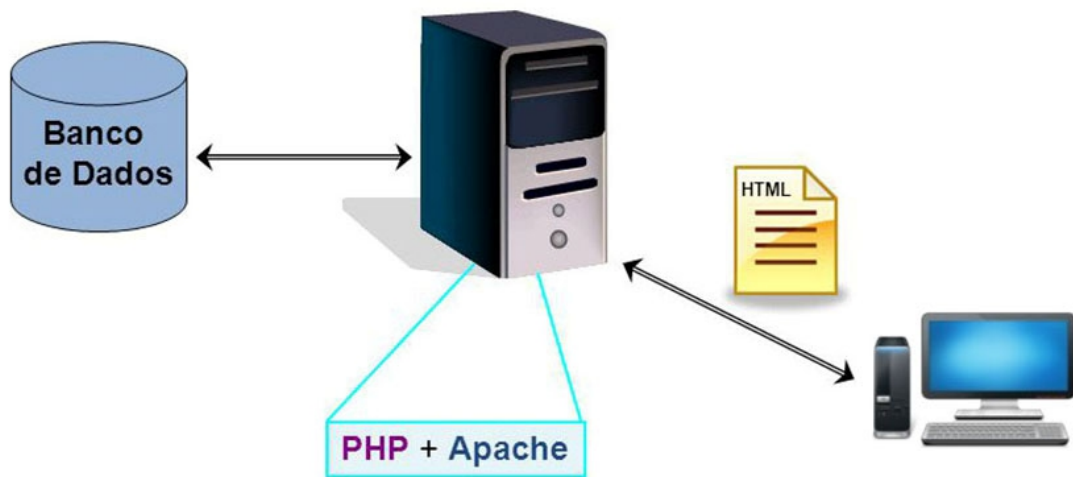
A Linguagem PHP

11

- Linguagem de Domínio Específico (web)
- Linguagem Interpretada (interpretador PHP)
- PHP é uma linguagem de programação *server-side* (roda no servidor)
- Linguagem Estruturada e Orientada a Objetos
- Tipagem
 - ▣ Fraca
 - ▣ Dinâmica
- Case-Sensitive

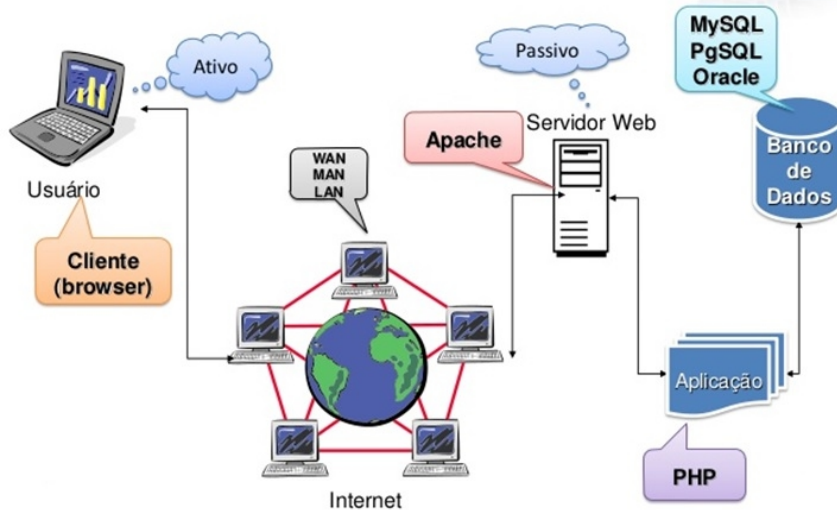
PHP – Arquitetura Cliente-Servidor

12



PHP – Arquitetura Cliente-Servidor

13



PHP – Arquitetura Cliente-Servidor

14

- ❑ Os comandos PHP são executados no servidor
- ❑ O cliente só recebe códigos HTML
- ❑ Não é possível copiar o código-fonte, diferente de HTML e JavaScript
- ❑ Site dinâmico, manipula códigos HTML através da linguagem PHP

Criando um servidor PHP

15

- Necessário:
 - ▣ Servidor Apache - <http://httpd.apache.org/download.cgi>
 - ▣ Servidor PHP - <http://www.php.net/downloads.php>
 - ▣ Servidor MySQL - <http://dev.mysql.com/downloads/>
- No entanto, vamos utilizar:
 - ▣ **XAMPP** - https://www.apachefriends.org/pt_br/download.html
 - Contém num só pacote o Apache, o PHP e o MySQL
 - ▣ Outra opção:
 - **WampServer** - <http://www.wampserver.com/en/>

16

Primeira página em PHP

PHP – Olá Mundo!!!

```
1. <?php
2.     echo("Olá Mundo!");
3. ?>
```

17

□ Em C++:

```
1. #include <iostream>
2. using namespace std;
3. int main() {
4.     cout << "Olá Mundo!";
5.     return 0;
6. }
```

□ Em C#:

```
1. using System;
2. namespace OlaMundo {
3.     class OlaMundo {
4.         static void Main() {
5.             Console.WriteLine("Olá Mundo!");
6.         }
7.     }
8. }
```

□ Em Java:

```
1. public class OlaMundo {
2.     public static void main(String[] args){
3.         System.out.println("Olá Mundo!");
4.     }
5. }
```

□ Em Pascal:

```
1. program OlaMundo;
2. begin
3.     writeln("Olá Mundo!");
4. end.
```

PHP – Estrutura

18

- **<?php** – Inicia os comandos PHP
- **//** - utilizado para fazer comentários de uma determinada linha
- **/* ... */** - Um bloco de comentários, podendo conter várias linhas.
- **echo()** – comando para escrever na tela, ele deve ser escrito em minúsculo
- **?>** - Finaliza o comando PHP
- Toda linha de comando finaliza com **;**

Primeira página em PHP

19

- "Olá Mundo!" em PHP:

```
<?php
// Comentários de códigos PHP
# Comentários de códigos PHP
echo("Olá Mundo!");
print("Olá Mundo!");
?>
```

Primeira página em PHP

20

- Primeira página em PHP – *Olá Mundo!!!*



Primeira página em PHP

21

1. Iniciar o servidor (Apache via XAMPP)

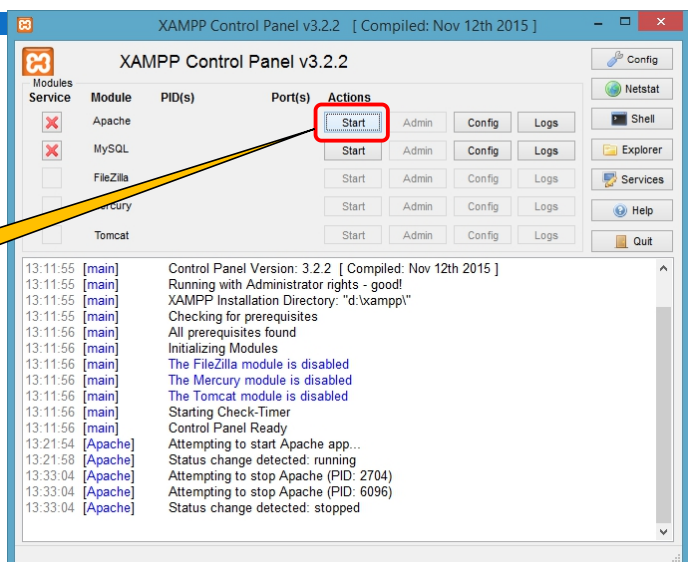
- Caso não estiver instalado baixe o **XAMPP**, que contém o Apache, PHP e MySQL - https://www.apachefriends.org/pt_br/download.html
- Tutorial - Como baixar e utilizar o XAMPP (Apache, MySQL e PHP) + ERROS COMUNS! - <https://www.youtube.com/watch?v=VhQk--V1934>

Primeira página em PHP

22

1. Iniciar o servidor (Apache via XAMPP)

Clique no botão 'Start' para iniciar o servidor.

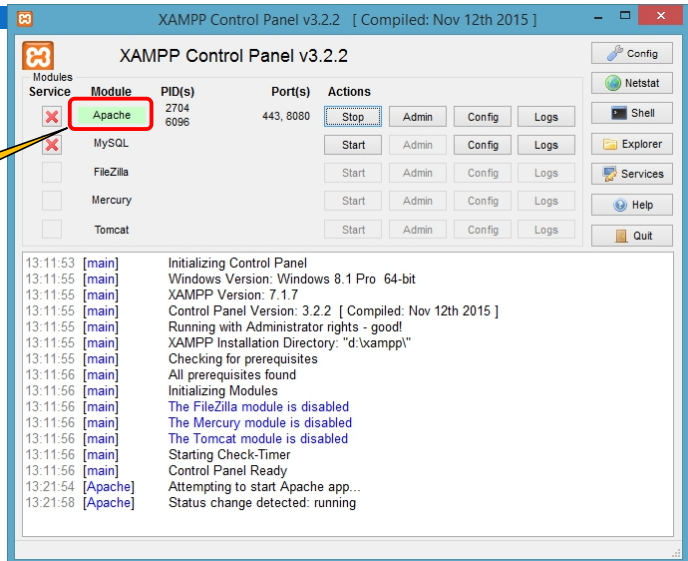


Primeira página em PHP

23

1. Iniciar o servidor (Apache via XAMPP)

Servidor iniciado corretamente...



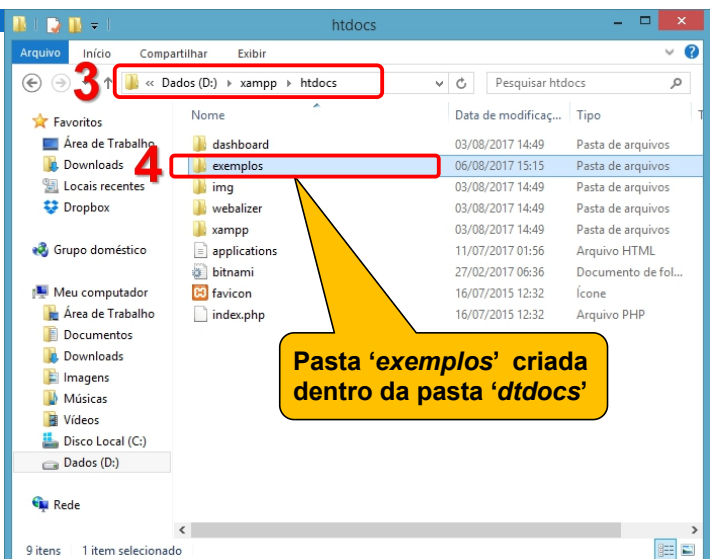
Primeira página em PHP

24

3. Acesse a '**htdocs**' pasta do XAMPP no PC:

- **C:\xampp\htdocs**

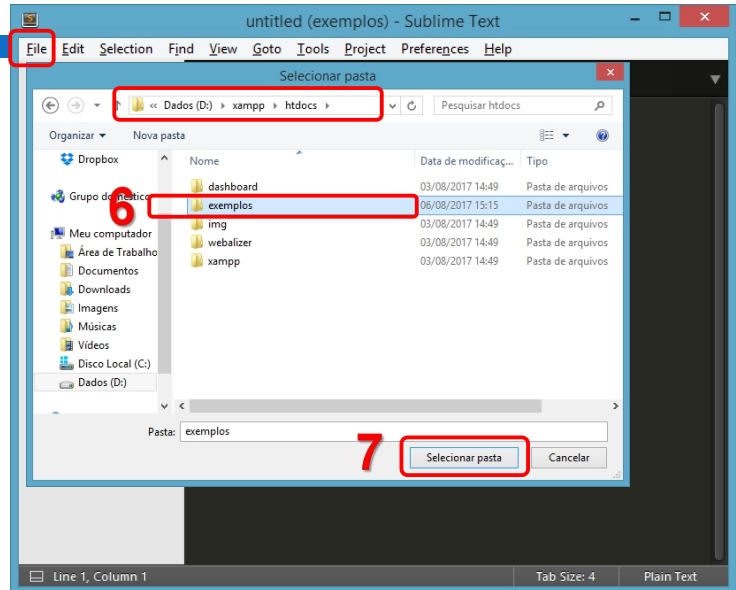
4. Crie uma pasta com o nome '**exemplos**'



Primeira página em PHP

25

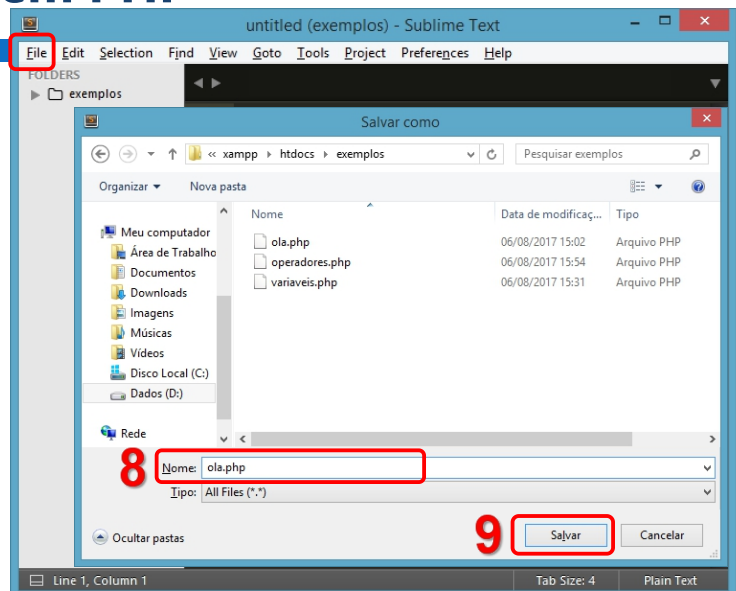
5. Abra o programa '**Sublime Text**' e acesse '**File → Open Folder...**'
6. Selecione a pasta '**exemplos**' criada anteriormente...
 - **Endereço:**
C:\xampp\htdocs
7. Clique em '**Selecionar pasta**'



Primeira página em PHP

26

8. Salve a página PHP com o nome '**ola.php**', dentro da pasta '**exemplos**' criada anteriormente:
 - '**File → Save As...**'
9. Clique em '**Salvar**'

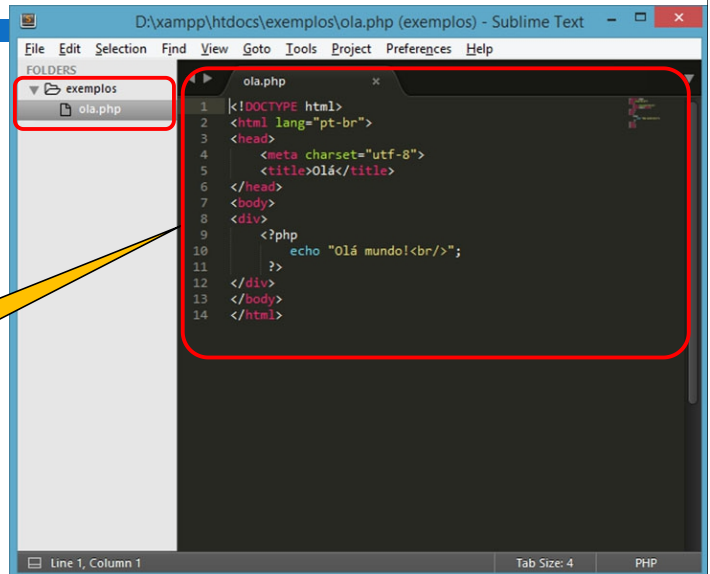


Primeira página em PHP

27

10. Implemente o código PHP na página...

Código HTML e PHP da página 'ola.php'



The screenshot shows a Sublime Text editor window titled 'D:\xampp\htdocs\exemplos\ola.php (exemplos) - Sublime Text'. The left sidebar shows a folder named 'exemplos' containing the file 'ola.php'. The main editor area displays the following code:

```
1 <!DOCTYPE html>
2 <html lang="pt-br">
3 <head>
4   <meta charset="utf-8">
5   <title>Olá</title>
6 </head>
7 <body>
8 <div>
9   <?php
10    echo "Olá mundo!<br/>";
11  ?>
12 </div>
13 </body>
14 </html>
```

Primeira página em PHP

28

11. No navegador, acesse o arquivo '**ola.php**' criado na pasta '**exemplos**'
 - Endereço: <http://localhost:8080/exemplos/ola.php>

Página 'ola.php' criada dentro da pasta 'D:\xampp\htdocs\exemplos' no servidor...



Sintaxe Básica da Linguagem PHP

PHP – Sintaxe

- ❑ Muito parecida com a linguagem C
- ❑ Tem suas particularidades
- ❑ Faz distinção entre letras maiúsculas e minúsculas (*Case-sensitive*)
- ❑ Instruções são expressões terminadas pelo símbolo ";" (ponto-e-vírgula)

PHP – Sintaxe

31

- A linguagem possui sintaxe e semânticas simples:
 - ▣ Comandos de atribuição;
 - ▣ Operadores;
 - ▣ Estruturas de controle;
 - ▣ Funções Pré-definidas.

PHP – Sintaxe

32

- Podemos definir e utilizar:
 - ▣ Variáveis;
 - ▣ Constantes;
 - ▣ Estruturas de dados mais complexas;
 - ▣ Funções

PHP – Sintaxe

33

- Saída - exibe no navegador o texto digitado
 - ▣ Exemplo: `echo("Texto digitado...");`

- Comentários
 - ▣ `/* ... */` - Um bloco de comentários, podendo conter várias linhas.
 - ▣ `//` Comenta apenas uma linha dessa maneira, caso o comentário passe dessa linha, ele causará uma falha.

PHP – Variáveis

34

- Não se declara o tipo da variável
- Pode-se atribuir um valor a uma variável de acordo com a execução da página
- O valor de uma variável é o seu último valor atribuído
- Sempre iniciam com o caractere "\$"
- Variáveis são atribuídas com o operador "="
- Diferencia caracteres maiúsculos de minúsculos
 - ▣ `$nome` é diferente de `$Nome`
- Quando as variáveis são apenas declaradas, elas possuem valor padrão (NULL ou ZERO).

PHP – Exemplos de Declaração de Variáveis

35

- ▣ *\$MinhaPrimeiraVariavel*
- ▣ *\$contador*
- ▣ *\$_x*
- ▣ *\$a1*
- ▣ *\$a_2* } **Atenção: Neste caso são criadas**
- ▣ *\$A_2* } **duas variáveis distintas**

- ▣ Saída:
 - ▣ *"echo(\$contador);"*: exibe no navegador o valor armazenado na variável

PHP – Tipos das Variáveis

36

- ▣ **Inteiro (*integer*)**: números inteiros, sem parte fracionária
 - ▣ Exemplo: *87*, *-34*, *0*, *1644*
- ▣ **Ponto Flutuante (*real*, *double* e *float*)**: números decimais
 - ▣ Exemplo: *6.54*, *-34.65*, *32.0*, *3.1415*
- ▣ **Cadeira de Caracteres (*string*)**: conjunto de caracteres
 - ▣ Exemplo: *'Maria'*, *'A'*, *'José da Silva'*
- ▣ **Lógico (*boolean*)**: tem dois possíveis valores: Verdadeiro (*True*) ou Falso (*False*) - 0 (falso), caso contrário (verdadeiro)
 - ▣ Exemplo: *True*, *False*

PHP – Tipos das Variáveis

37

- **Arrays:** são coleções identificadas e indexadas de outros valores
 - ▣ Exemplo: `$cores = array("vermelho", "verde", "azul");`
- **Objetos:** são instâncias de classes definidas pelo programador, as quais podem empacotar tanto outros tipos de valores como funções que são específicas à classe.
 - ▣ Exemplo: `$pessoa = new Pessoa();`

PHP – Variáveis – exemplos

38

```
1. <?php
2.     $idade = 23; // Variável Inteira
3.     $salario = 998.85; // Variável Decimal
4.     $nome = "Maria"; // Variável caractere
5.     $casada = true; // Variável booleana
6.     $filhos = array("Pedro", "Ana", "Paulo"); // Array
7.     echo($nome." tem ". $idade." anos e ganha R$ ".$salario);
8.     echo("<br>".$nome." é casada? ".$casada);
9. ?>
```

Obs.: Concatenação de *Strings*: usa-se o ponto (.)

PHP – Tipos das Variáveis

39

- Para saber qual é o tipo de uma determinada variável utilize a função **gettype()**:

- Exemplo:

```
1. $nome = 'Maria';  
2. gettype($nome);
```

- Este exemplo, a função **gettype()** exibirá na tela a seguinte mensagem:

string

PHP – Tipos das Variáveis

40

- **Função gettype()**

- ▣ Retorna o tipo de uma variável
- ▣ Exemplo:

EXEMPLO	RETORNO
gettype('Olá');	string
gettype(30);	integer
gettype(40.58);	double

PHP – Tipagem das Variáveis

41

- As variáveis em PHP possuem **tipagem fraca** e **dinâmica**
 - ▣ **Dinamicamente Tipada**: Quando se atribui um valor para uma variável, esta variável se torna do tipo do valor atribuído, podendo ser alterado o tipo desta variável caso for atribuído novo valor de outro tipo de dado;
 - ▣ **Fracamente Tipada**: Caso for solicitado uma operação de um tipo diferente do tipo da variável, a variável será convertida implicitamente para o tipo mais genérico (Coerção).

Transformação de tipos de variáveis por coerção

42

- **Coerção**
 - ▣ é a conversão de um tipo em outro tipo diferente mediante operação realizada com tipos diferentes. Exemplos:

```
1. <?php
2.   $a = 5; // Variável Inteira
3.   $a = $a + '5'; // Variável String
4.   $pi = 3;
5.   $pi = $pi + 0.14159; // Variável Decimal
6. ?>
```

Nesse exemplo a variável **\$a** é numérica (*integer*) na primeira atribuição e *string* na segunda atribuição.

Transformação explícita de tipos de variáveis

43

- Feita via ***typecast***

```
1.<?php
2.    $a = 6; // Variável Inteira (6)
3.    $a = (float)6; // Variável Float (6.0)
4. ?>
```

Nesse exemplo a variável **\$a** é numérica (*integer*) na primeira atribuição e *float* na segunda atribuição.

Transformação explícita de tipos de variáveis

44

- Feita via ***typecast***

```
1.<?php
2.    $num1 = 10.5; // Variável Real
3.    $num2 = 8; // Variável Inteira
4.    $resultado = (int)$num1 + $num2;
5.    echo("O resultado da soma é ". $resultado);
6. ?>
```

Nesse exemplo a variável **\$resultado** recebe um número inteiro (18).

Operadores em PHP

- **Atribuição**
- **Aritméticos**
- **Unários**
- **Relacionais**
- **Lógicos**

PHP – Operadores de Atribuição

- Atribui um valor a uma determinada variável

OPERADOR	OPERAÇÃO	EXEMPLO	É O MESMO QUE
=	Atribui	<code>\$x = 5</code>	<code>\$x = 5</code>
+=	Soma e Atribui	<code>\$x += 5</code>	<code>\$x = \$x + 5</code>
-=	Subtrai e Atribui	<code>\$x -= 5</code>	<code>\$x = \$x - 5</code>
*=	Multiplica e Atribui	<code>\$x *= 5</code>	<code>\$x = \$x * 5</code>
/=	Divide e Atribui	<code>\$x /= 5</code>	<code>\$x = \$x / 5</code>
%=	Módulo e Atribui	<code>\$x %= 5</code>	<code>\$x = \$x % 5</code>
.=	Concatena e Atribui	<code>\$a .= 'Olá'</code>	<code>\$a = \$a . 'Olá'</code>

PHP – Operadores de Atribuição

47

- PHP aceita a atribuição múltipla.
- Exemplo:

```
$a = $b = $c = 5; // $a=5; $b=5; $c=5;
```

PHP – Operadores de Atribuição

48

- Exemplos:

```
1. $mais_igual = $menos_igual = $vezes_igual =  
   $dividido_igual = $modulo_igual = 9;  
  
2. $mais_igual += 1; // resultado 10  
3. $menos_igual -= 1; // resultado 8  
4. $vezes_igual *= 1; // resultado 9  
5. $dividido_igual /= 1; // resultado 9  
6. $modulo_igual %= 2; // resultado 1
```


PHP – Operadores Aritméticos

49

- Realiza uma determinada operação entre dois valores.

OPERADOR	OPERAÇÃO	EXEMPLO	RESULTADO
+	Adição	8 + 2	10
-	Subtração	8 - 2	6
*	Multiplicação	8 * 2	16
/	Divisão	8 / 2	4
%	Resto da Divisão	8 % 2	0

PHP – Operadores Aritméticos – Exemplos

50

- Adição (caractere: +)
 - ▣ *\$adicao = \$num1 + \$num2;*
- Subtração (caractere: -)
 - ▣ *\$subtracao = \$num1 - \$num2;*
- Multiplicação (caractere: *)
 - ▣ *\$multiplicacao = \$num1 * \$num2;*
- Divisão (caractere: /)
 - ▣ *\$divisao = \$num1 / \$num2;*
- Resto da divisão (caractere: %)
 - ▣ *\$restodivisao = \$num1 % \$num2;*

PHP – Operadores Aritméticos – Exemplos

51

```
1.<?php
2.  $num1 = 8;
3.  $num2 = 2;
4.  $soma = $num1 + $num2; // Operador de adição
5.  echo("<br>A soma de ".$num1." + ".$num2." é: ".$soma);
6.  $media = $num1 + $num2 / 2; // Operador de adição e de divisão
7.  echo("<br>A média de ".$num1." e ".$num2." é: ".$media);
8.?>
```

PHP – Operadores Aritméticos

52

- Precedência de operadores
 - ▣ Baseado nos conceitos da matemática

ORDEM	OPERAÇÃO	SÍMBOLOS
1ª	Parênteses	()
2ª	Multiplicação, divisão, resto da divisão	*, /, %
3ª	Adição e subtração	+, -

PHP – Operadores Unários

53

- ++ Incremento
- -- Decremento
- ++\$a - pré-incremento (incrementa \$a em um e depois retorna \$a)
- \$a++ - pós-incremento
- --\$a - pré-decremento
- \$a-- - pós-decremento

PHP – Operadores Unários

54

- PHP - Operador de incremento (++)

```
1. <?php
2.     $x = 23;
3.     $y = $x++; // Operador de pós-incremento
4.     echo($x." e ".$y); // 24 e 23

5.     $x = 23;
6.     $y = ++$x; // Operador de pré-incremento
7.     echo($x." e ".$y); // 24 e 24
8. ?>
```

PHP – Operadores Relacionais

- Compara dois valores e retorna um valor lógico (*True* ou *False*).

OPERADOR	OPERAÇÃO	EXEMPLO	RESULTADO
==	É igual a	5 == 8	False
!=	Não é igual	5 != 8	True
>	É maior que	5 > 8	False
<	É menor que	5 < 8	True
>=	É maior que ou igual a	5 >= 8	False
<=	É menor que ou igual a	5 <= 8	True

PHP – Operadores de Igualdade e de Atribuição

57

- Não confunda o operador de igualdade "**==**" com o operador de atribuição "**=**"
 - ▣ A expressão "**\$x == \$y**" compara se as variáveis **\$x** e **\$y** têm os dois valores idênticos e, se for idênticos, retorna **True**, ou se não for idênticos retorna **False**.
 - ▣ Já a expressão "**\$x = \$y**" atribui o valor da variável **\$y** a variável **\$x**.

PHP – Operadores Lógicos

58

- Compara duas expressões e retorna um valor lógico.
 - ▣ Exemplo: **\$x = 6; \$y = 3;**

OPERADOR	OPERAÇÃO	EXEMPLO	RESULTADO
and	E	((x < 10) and (y > 1))	True
&&	E	((x < 10) && (y > 1))	True
or	Ou	((x == 5) or (y == 5))	False
 	Ou	((x == 5) (y == 5))	False
not	Negação	not (x == y)	True
!	Negação	! (x == y)	True

PHP – Estruturas de Controle

59

- Estruturas de Controle em PHP
 - ▣ **Blocos de códigos sequenciais**
 - ▣ **Estruturas condicionais**
 - ▣ **Estruturas de repetição**
 - ▣ **Funções**
- Estas estruturas são agrupadas por abre e fecha chaves (**{** e **}**)

PHP – Blocos de códigos

60

- **Blocos de códigos PHP**
 - ▣ É possível começar e terminar o bloco de código PHP quantas vezes desejar...

PHP – Blocos de códigos

61

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Bloco de Códigos PHP</title>
  </head>
  <body>
    <?php
      $num1 = 60;
      $num2 = 70;
    ?>
    <?php
      $media = ($num1 + $num2) / 2;
      $mensagem = "A média é de $num1 e $num2 é: ".$media;
    ?>
    <h2> <?php echo($mensagem); ?> </h2>
  </body>
</html>
```

PHP – Blocos de códigos

62

```
<!DOCTYPE html>
<html lang="pt-br">
  <head>
    <title>Bloco de Códigos PHP</title>
  </head>
  <body>
    <?php
      $num1 = 60;
      $num2 = 70;
    ?>
    <?php
      $media = ($num1 + $num2) / 2;
      $mensagem = "A média é de $num1 e $num2 é: ".$media;
    ?>
    <h2> <?php echo($mensagem); ?> </h2>
  </body>
</html>
```

1º bloco de códigos

2º bloco de códigos

3º bloco de códigos

PHP – Estruturas Condicionais - IF

63

- Sintaxe:

```
if (<condição>)  
    <comando>;  
  
if (<condição>) {  
    <comando 1>;  
    ...  
    <comando N>;  
}
```

PHP – Estruturas Condicionais – IF-ELSE

64

- Sintaxe:

```
if (<condição>)  
    <comando 1>;  
else  
    <comando 2>;
```

```
if (<condição>) {  
    <comando 1>;  
    ...  
    <comando N>;  
} else {  
    <comando 2>;  
    ...  
    <comando N>;  
}
```


PHP – Estruturas Condicionais – IF-ELSE IF-ELSE

65

- Sintaxe:

```
if (<condição 1>) {  
    <comando 1>;  
}  
  
else if (<condição 2>) {  
    <comando 2>;  
}  
  
else {  
    <comando N>;  
}
```

PHP – Estruturas Condicionais – exemplo IF

66

```
<!DOCTYPE html>  
<html lang="pt-br">  
  <head> <title>Estrutura Condicional - IF</title> </head>  
  <body>  
    <?php  
      $num1 = 60; $num2 = 70;  
      $media = ($num1 + $num2) / 2;  
      if ($media >= 60) {  
        echo("Aprovado");  
      } else if (($media >= 40) && ($media < 60)) {  
        echo("Prova final");  
      } else {  
        echo("Reprovado");  
      }  
    ?>  
  </body>  
</html>
```

PHP – Estruturas Condicionais – SWITCH

67

- Sintaxe:

```
switch (<expressão>) {  
    case <valor 1>:  
        <comando A>;  
    break;  
    case <valor 2>:  
        <comando B>;  
    break;  
    default:  
        <comando(s) default>;  
    break;  
}
```

PHP – Estruturas Condicionais – exemplo SWITCH

68

```
<!DOCTYPE html>  
<html lang="pt-br">  
  <head> <title>Estrutura Condicional - SWITCH</title> </head>  
  <body>  
    <?php  
      $i = 2;  
      switch($i) {  
        case 1:  
          echo("Janeiro");  
          break;  
        case 2:  
          echo("Fevereiro");  
          break;  
        default:  
          echo("Mês não encontrado.");  
          break;  
      } ?>  
  </body> </html>
```

PHP – Operador Ternário (ou *if ternário*)

69

□ `<expressão 1> ? <expressão 2> : <expressão 3>;`

□ Exemplo:

```
$a = 3;  
$b = ($a > 5 ? "É maior" : "Não é maior");
```

□ é equivalente a:

```
if ($a > 5) {  
    $b = "É maior";  
} else {  
    $b = "Não é maior"; }  

```

PHP – Operador Ternário (ou *if ternário*)

70

```
<!DOCTYPE html>  
<html lang="pt-br">  
  <head>  
    <title>Operador Ternário</title>  
  </head>  
  <body>  
    <?php  
      $a = 3; // Variável inteira  
      $b = ($a > 5 ? "É maior" : "Não é maior");  
      echo($b);  
    ?>  
  </body>  
</html>
```

PHP – Estruturas de Repetição

- ***while*** - ***do .. while***
 - ▣ Usado quando não é possível definir o número exato de repetições.
- ***for***
 - ▣ Usado quando sabe exatamente a quantidade de repetições de um trecho de código.
- ***foreach***
 - ▣ Usado para percorrer uma lista ou um *array*.

PHP – Estruturas de Repetição – WHILE

73

- Sintaxe:

```
while (condição) {  
    <comando>;  
}
```

```
while (condição) {  
    <comando 1>;  
    ...  
    <comando N>;  
}
```

PHP – Estruturas de Repetição – WHILE

74

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Estrutura de Repetição - WHILE</title>  
  </head>  
  <body>  
    <?php  
      $i = 1;  
      while ($i < 5) {  
        echo($i);  
        i++;  
      }  
    ?>  
  </body>  
</html>
```

PHP – Estruturas de Repetição – DO .. WHILE

75

- Sintaxe:

```
do {  
    <comando>;  
} while (condição);  
  
do {  
    <comando1>;  
    ...  
    <comandoN>;  
} while (condição);
```

PHP – Estruturas de Repetição – DO .. WHILE

76

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Estrutura de Repetição – DO .. WHILE</title>  
  </head>  
  <body>  
    <?php  
      $acabou = false;  
      do {  
        echo("<br>loop...");  
        $acabou = true;  
      } while (!$acabou);  
    ?>  
  </body>  
</html>
```

PHP – Estruturas de Repetição – FOR

77

- Sintaxe:

```
for (valor inicial; condição; incremento) {  
    <comando1>  
    ..  
    <comandoN>  
}
```

PHP – Estruturas de Repetição – FOR

78

```
<!DOCTYPE html>  
<html>  
  <head>  
    <title>Estrutura de Repetição - FOR</title>  
  </head>  
  <body>  
    <?php  
      for ($i = 0; $i < 10; $i++) {  
        echo($i);  
      }  
    ?>  
  </body>  
</html>
```

Array – Conceito

- Armazena em uma única variável um conjunto de valores. Esses valores são acessados através de um índice.
 - ▣ Os índices ficam dentro de colchetes
- Um array no PHP é um mapa ordenado. Um mapa é um tipo que relaciona valores a chaves. Pode ser tratado como:
 - ▣ Um vetor, uma matriz, uma lista;
 - ▣ Uma *hashtable* (que é uma implementação de mapa)
 - ▣ Um dicionário, coleção, pilha, fila;
- Os valores do *array* podem ser outros *arrays*, árvores e *arrays* multidimensionais.

PHP – Array

81

□ Sintaxe Básica em PHP:

1. *// Cria / instancia o array sem nenhum elemento*
2. `$nome_do_array = array();`
3. *// Cria / instancia o array e já armazena um conjunto de valores (elementos)*
4. `$nome_do_array = array(valor1, valor2, ..., valorN);`
5. *// Retorna o valor armazenado 2ª posição do Array*
6. `$variavel = $nome_do_array[1];`
7. *// Atribui um novo valor à 2ª posição do Array*
8. `$nome_do_array[1] = novo_valor;`

PHP – Array

82

```
<!DOCTYPE html>
<html>
  <head>
    <title>PHP – Array</title>
  </head>
  <body>
    <?php
      $cores = array("vermelho", "verde", "azul");
      $cor = $cores[1]; // retorna "verde" (2º elemento do array)
      $cores[1] = "amarelo"; // atribui novo valor (ao 2º elemento do array)
    ?>
  </body>
</html>
```

PHP – *Array*

83

- Podem ser definidos como mapeamentos ou vetores indexados
- Exemplo:

```
1. <?php
2.     $cores[0] = "Vermelho";
3.     $cores[1] = "Verde";
4.     $cores[2] = "Azul";
5. ?>
```

PHP – *Array* – tipos de índices

84

- **Ordenado**
 - ▣ baseado em número (começa no 0) (indexada numericamente)
- **Associativo**
 - ▣ formado por caracteres alfanuméricos (indexada por nome)

PHP – Array – índice ordenado

85

```
1. <?php
2.     // cria e inicializa um array (indexada numericamente)
3.     $cores = array("vermelho", "verde", "azul");
4.
5.     OU
6.
7.     // cria e inicializa um array usando índices (explicitamente)
8.     $cores = array(0=>"vermelho", 1=>"verde", 2=>"azul");
9.
10.    OU
11.
12.    // cria e inicializa um array usando índices (numéricos)
13.    $cores[] = "Vermelho";
14.    $cores[] = "Verde";
15.    $cores[] = "Azul";
16.    ?>
```

PHP – Array – índice associativo

86

- São conjunto ordenados de chaves e valores, onde cada valor é acessado através de uma chave associada.
- Exemplo:

```
1. <?php
2.     $estados_e_capitais = array (
3.         "SP" => "São Paulo",
4.         "MG" => "Belo Horizonte",
5.         "RJ" => "Rio de Janeiro",
6.         "ES" => "Vitória"
7.     );
8.    ?>
```

PHP – Arrays Bidimensionais (matriz)

87

- Consiste em um *array* de duas dimensões.
- Exemplo:

```
1. <?php
2.     $nome["sala 1"][1] = "Antônio Carlos";
3.     $nome["sala 1"][2] = "Bruno José";
4.     $nome["sala 1"][3] = "Camila Miranda";
5.
6.     $nome["sala 2"][1] = "Marcos Cesar";
7.     $nome["sala 2"][2] = "Paulo Barbosa";
8.     $nome["sala 2"][3] = "Everton Jose";
9. ?>
```

PHP – Arrays – Funções

88

- **unset(\$array)** – “destrói” o *array*
- **count(\$array)** e **sizeof(\$array)** – ambos retorna a quantidade de elementos de um *array*
- **array_push(\$array, \$valor)** – insere um novo elemento no fim de um *array*
- **array_unshift (\$array, \$valor)** – insere um novo elemento no início de um *array*
- **array_pop(\$array)** – retira e retorna o último elemento do *array*
- **array_shift(\$array)** – retira e retorna o primeiro elemento de um *array*

PHP – Percorrendo um *Array*

89

- ❑ Percorrer um *array* utilizando a estrutura de repetição **FOR...**
- ❑ Exemplo:

```
1. <?php
2.     $cores = array("vermelho", "verde", "azul");
3.     for ($i = 0; $i < count($cores); $i++) {
4.         echo($cores[$i]."<br>");
5.     }
6.     ?>
```

PHP – Percorrendo um *Array*

90

- ❑ Percorrer um *array* utilizando a estrutura de repetição **FOREACH...**
 - ❑ **Estrutura FOREACH:** Usado para percorrer uma lista ou um *array*.
 - ❑ Exemplo:

```
1. <?php
2.     $cores = array("vermelho", "verde", "azul");
3.     foreach ($cores as $valor) {
4.         echo($valor."<br>");
5.     }
6.     ?>
```

PHP – *Array*

91

- Um *Array* em PHP pode ter elementos de tipos diferentes.
 - ▣ Exemplo:

```
1. <?php
2.     $pessoa = array("Maria", 23, "F", 68.71);
3.     $pessoa2 = array("Nome" => "Maria", "Idade"
                        => 23, "Sexo" => "F", "Peso" => 68.71);
4. ?>
```

92

Funções em PHP

Funções em PHP – Conceito

93

- São úteis para que não haja repetição de partes de códigos.
- Com as funções conseguimos um sistema ou página mais modular.
- Para manutenção também é interessante, pois alterando a função, todo o sistema que utiliza da função está alterado.

Funções em PHP – Variáveis e Funções

94

- Uma variável quando atribuímos valor dentro de uma função, não conseguimos ler o seu valor fora da função.
- Também não conseguimos ler uma variável que atribuímos valor fora de uma função dentro da função.

Funções em PHP – Estrutura

95

- ❑ Sempre inicia com a palavra ***'function'***;
- ❑ Nome da função, seguido de parênteses;
- ❑ Dentro dos parênteses temos os argumentos, não são obrigatórios;
- ❑ Depois temos as chaves que inicia e fecha o bloco de comando;
- ❑ Pode ou não retornar um valor.
- ❑ Sintaxe em PHP:

```
function nome_da_função([arg1, arg2, argN]) {  
    <comandos>;  
    [return <valor de retorno>];  
}
```

PHP – Funções sem argumentos / parâmetros

96

- ❑ Exemplo:

```
1. <?php  
2.     function ola() {  
3.         echo("Olá Mundo!!!");  
4.     }  
5.     ola(); // Chamando a função  
6. ?>
```


PHP – Funções com argumentos passados por valor

97

□ Exemplo:

```
1. <?php
2.     function quadrado($num) {
3.         echo("O quadrado de $num é: ". ($num * $num));
4.     }
5.     quadrado(3); // Chamando a função com argumento
6. ?>
```

PHP – Funções com Argumentos e Retorno

98

□ Exemplo:

```
1. <?php
2.     function retornaQuadrado($num) {
3.         $quadrado = ($num * $num);
4.         return $quadrado;
5.     }
6.     $quad = retornaQuadrado(3);
7.     echo("O quadrado de 3 é: ". $quad);
8. ?>
```

PHP – Funções em com argumentos passados por referência

99

□ Exemplo:

Parâmetro passado por referência.

```
1. <?php
2.   function dobro(&$num) {
3.       $num = ($num * 2);
4.   }
5.   $x = 5; # Variável $x recebe o valor 5
6.   dobro($x); # Função que recebe um argumento como referência
7.   echo("O dobro é: ". $x); # Imprime o valor 10
8. ?>
```

PHP – Funções com valores padrão de argumentos

100

□ Exemplo:

Argumento com valor padrão.

```
1. <?php
2.   function fazerCafe($tipo = "puro") {
3.       return "Fazendo uma xícara de café ". $tipo;
4.   }
5.   echo(fazerCafe()); # Imprime Fazendo uma xícara de café puro
6.   echo(fazerCafe(null)); # Imprime Fazendo uma xícara de café
7.   # Imprime Fazendo uma xícara de café expresso
8.   echo(fazerCafe("expresso"));
9. ?>
```

PHP – Funções com valores padrão de argumentos

101

- Uso **incorreto** de parâmetros padrão de função
- Exemplo:

Os argumentos opcionais devem ser os últimos parâmetros.

```
1. <?php
2. function fazerSuco($sabor="sem açúcar", $fruta) {
3.     return "Suco de $fruta $sabor";
4. }
5. echo(fazerSuco("laranja")); # Erro!
6. ?>
```

PHP – Funções com valores padrão de argumentos

102

- Uso **correto** de parâmetros padrão de função
- Exemplo:

Os argumentos opcionais devem ser os últimos parâmetros.

```
1. <?php
2. function fazerSuco($fruta, $sabor="sem açúcar") {
3.     return "Suco de $fruta $sabor";
4. }
5. echo(fazerSuco("laranja")); # Funciona corretamente
6. ?>
```

PHP – Funções número variável de argumentos

103

- Os argumentos são passados na forma de um *array*. Exemplo:

```
1. <?php
2. function somar( ...$numeros ) {
3.     $soma = 0;
4.     foreach ( $numeros as $n ) {
5.         $soma = $soma + $n;
6.     }
7.     return $soma;
8. }
9. echo(somar(1, 2, 3, 4)); # Imprime 10
10. ?>
```

Vários valores
passados como
argumentos.

Percorre cada elemento
do arry passado como
parâmetro.

104

Variáveis Locais e Globais em PHP

Variáveis Locais e Globais em PHP

105

- Analise o código seguinte...

```
1. <?php
2.   $a = 5; // Variável global
3.   $b = 3; // Variável global
4.   function somar() {
5.       $soma = ($a + $b);
6.   }
7.   somar(); // Chama a função
8.   echo($soma); // Imprime o resultado
9. ?>
```

Variáveis Locais e Globais em PHP

106

- **Por que não funcionou?**
 - ▣ As variáveis **\$a**, **\$b** declaradas fora da função são variáveis globais e as variáveis **\$soma**, **\$a**, **\$b** declaradas dentro da **function somar()** são variáveis locais, ou seja, são variáveis totalmente diferentes.
- Mas como a função reconhece as variáveis globais?
 - ▣ Existem duas formas:

Variáveis Locais e Globais em PHP

107

- 1ª forma – Variáveis globais dentro de uma função em PHP:

```
1. <?php
2.     $a = 5; // Variável global
3.     $b = 3; // Variável global
4.     function somar() {
5.         // "Avisa" a função que estas variáveis são variáveis globais...
6.         global $a, $b, $soma;
7.         $soma = ($a + $b);
8.     }
9.     somar(); // Chama a função
10.    echo($soma); // Imprime o resultado
11. ?>
```

Variáveis Locais e Globais em PHP

108

- 2ª forma – Variáveis globais dentro de uma função em PHP:

```
1. <?php
2.     $a = 5; // Variável global
3.     $b = 3; // Variável global
4.     function somar() {
5.         // "Avisa" a função que estas variáveis são variáveis globais...
6.         $GLOBALS["soma"] = ($GLOBALS["a"] + $GLOBALS["b"]);
7.     }
8.     somar(); // Chama a função
9.     echo($soma); // Imprime o resultado
10. ?>
```

Dúvidas?

109



Prof. Me. Fernando Roberto Proença

fernandorroberto@gmail.com

Referências

110

- Tutorial - Como baixar e utilizar o XAMPP (Apache, MySQL e PHP) + ERROS COMUNS! - <https://www.youtube.com/watch?v=VhQk--V1934>

Cursos / Tutoriais interessantes

111

- ❑ [Manual do PHP](#)
- ❑ [PHP - W3Schools](#)
- ❑ [PHP para Iniciante - Curso em Vídeo](#)
- ❑ [POO com PHP - Curso em Vídeo](#)
- ❑ [PHP - do jeito certo](#)