

Tecnologias server-side: PHP

Carlos Santos . NTC . DeCA . UA

Aula T11, 23 mar 2020

um pouco de história

► 1995

- **Rasmus Lerdorf** desenvolve Scripts CGI em Perl (contador de visitantes da sua página)
- oferece esses script sob o nome – **Personal Home Page**



► 1997

- **PHP 2.0** (**P**ersonal **H**ome **P**age/**F**orm **I**nterpreter) - utiliza linguagem C
- popularidade cresce exponencialmente
- desenvolve-se uma comunidade de programadores/utilizadores

► 1998

- **PHP 3.0**, em 1999 já tem mais de 1 milhão de utilizadores
- **Andi Gutmans**, **Zeev Suraski**



um pouco de história

► 2000

- **PHP 4.0** passa a significar PHP: Hypertext Preprocessor
- motor/parser rescrito -> Zend Engine
- introduz: gestão recursos, OOP e gestão de sessões

► 2004

- **PHP 5.0**
- maior suporte a OOP com desenvolvimento de frameworks (Zend)
- suporte a XML, Web Services,...
- utilizada em mais de 20 milhões de web sites

► 2015 (dezembro)

- **PHP 7.0** (performance e gestão de recursos)

performance

Campus by Fundação Altice com PHP 5.6

<input type="checkbox"/>	ChIJSTCO...	2...	x...	<u>clien...</u>	3...	494 ms	
<input type="checkbox"/>	ChIJ78_R...	2...	x...	<u>clien...</u>	4...	553 ms	
<input type="checkbox"/>	ChIJN0njl...	2...	x...	<u>clien...</u>	2...	796 ms	
<input type="checkbox"/>	ChIJj1n28...	2...	x...	<u>clien...</u>	2...	604 ms	
<input type="checkbox"/>	ChIJSTCO...	2...	x...	<u>clien...</u>	3...	715 ms	

Campus by Fundação Altice com PHP 7.0 (logo no início do lançamento)

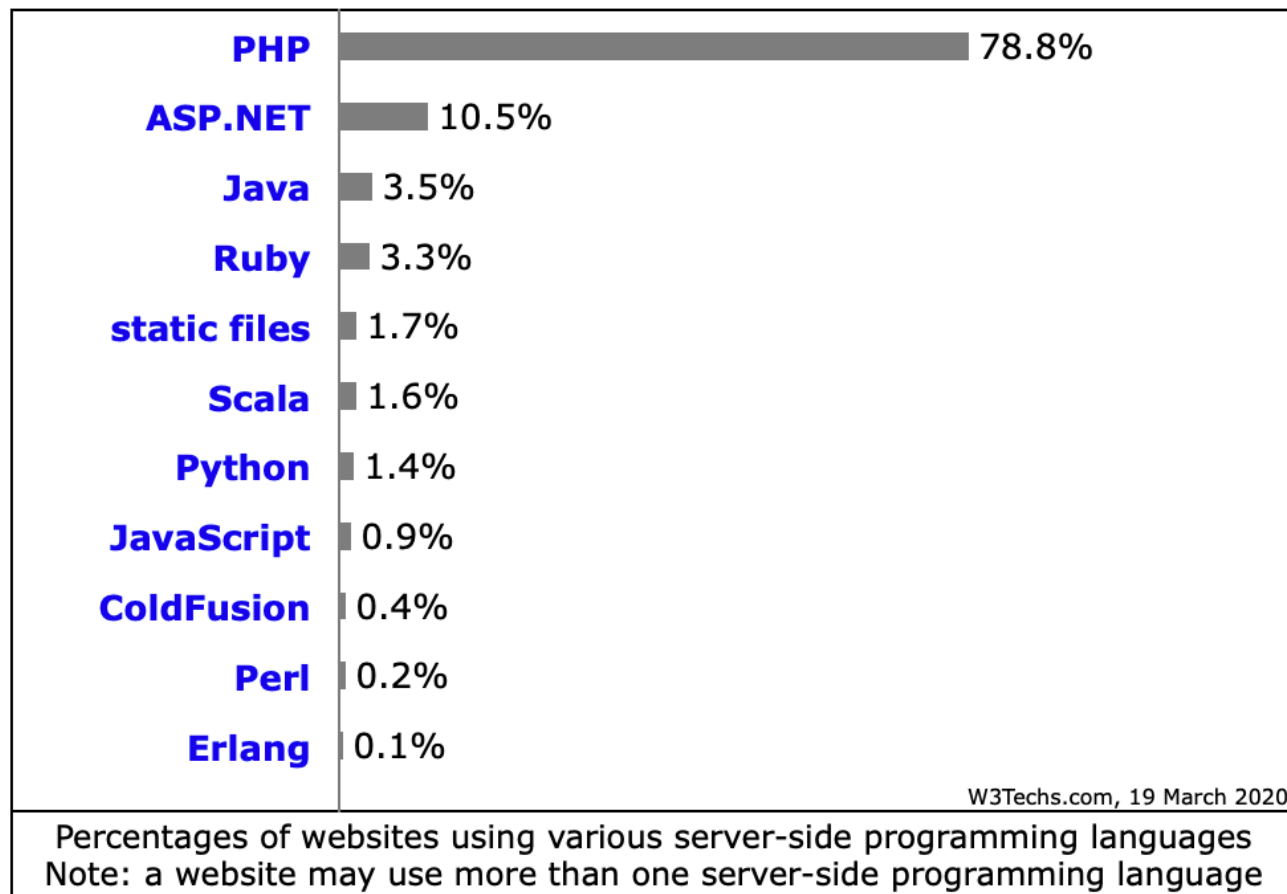
<input type="checkbox"/>	ChIJK9ob...	2...	x...	<u>clien...</u>	3...	222 ms	
<input type="checkbox"/>	ChIjk8Gy...	2...	x...	<u>clien...</u>	3...	198 ms	
<input type="checkbox"/>	ChIjk8Gy...	2...	x...	<u>clien...</u>	3...	147 ms	
<input type="checkbox"/>	ChIjk8Gy...	2...	x...	<u>clien...</u>	3...	169 ms	
<input type="checkbox"/>	ChIjk8Gy...	2...	x...	<u>clien...</u>	3...	158 ms	
<input type="checkbox"/>	ChIjk8Gy...	2...	x...	<u>clien...</u>	3...	155 ms	

Usage statistics of server-side programming languages for websites

This diagram shows the percentages of websites using various server-side programming languages. See [technologies overview](#) for explanations on the methodologies used in the surveys. Our reports are updated daily.

How to read the diagram:

PHP is used by 78.8% of all the websites whose server-side programming language we know.



arquitetura

- ▶ O PHP funciona numa plataforma Web
 - ▶ exemplo: Apache Web Server + Módulo PHP
- ▶ É incluído numa estrutura em 3 camadas

Apresentação (client-side)

- HTML, CSS, Javascript, AJAX, ...

Lógica (server-side)

- PHP

Dados (server-side)

- MySQL

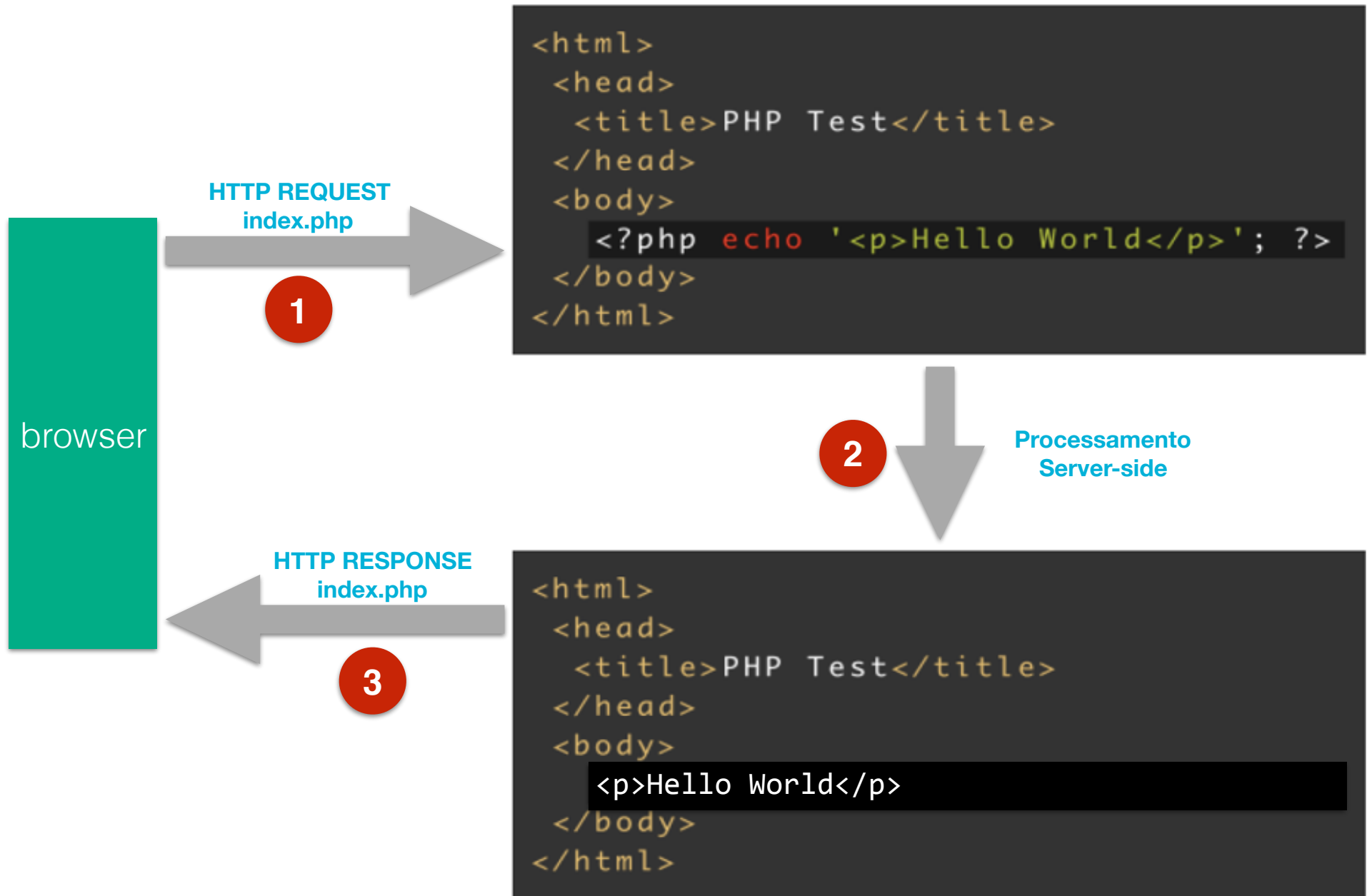
- ▶ a linguagem PHP é processada no servidor (server-side)
- ▶ os scripts podem estar embebidos em páginas HTML
- ▶ normalmente serve para gerar HTML

PHP: algumas características

- ▶ **simples de usar**, funções mais requisitadas estão disponível por defeito
- ▶ **poderosa**, centenas de bibliotecas disponíveis
- ▶ **flexível**, permite a livre escolha das soluções para a implementação das aplicações Web (por exemplo: escolher livremente o SGBDR a utilizar)
- ▶ **gratuita**
- ▶ **desenvolvida colaborativamente**, maior evolução (mas também gera algumas dificuldades)



funcionamento geral



páginas estáticas

- ▶ Uma **página estática** é uma página cujo **conteúdo e aspeto foram totalmente determinados pelo seu autor**, no momento em que escreveu o código.
- ▶ o conteúdo e o aspeto são constantes independentemente de **quem** visita a página, **quando** a visita ou **como** a visita
- ▶ alterações numa página estática só podem ser efectuadas através da edição do código
 - ▶ exemplo: uma página HTML que se encontra publicada num servidor Web
 - ▶ nota: não confundir este conceito com elementos interativos numa página Web que são controlados através de Javascript

páginas dinâmicas

- ▶ Uma **página dinâmica** para a Web é uma página cujo **conteúdo e aspeto não são totalmente determinados no seu estado inicial** (isto é, quando foi publicada pelo seu autor)
 - ▶ os conteúdos e aspeto são determinados após um utilizador executar um pedido da página ao servidor Web
 - ▶ os conteúdos e aspeto da página podem variar de pedido para pedido
 - ▶ exemplo: página HTML com PHP embebido
 - ▶ no momento em que existe um pedido de acesso à página, o PHP embebido é executado no servidor para gerar a página final

páginas dinâmicas

► Tecnologias server-side (executado no servidor)

- PHP
- .Net (C#, ASP.Net)
- JSP
- Perl
- Python
- Ruby
- NodeJS



► Tecnologias client-side (executado no browser)

- JavaScript
- Dart

versão do PHP

► <?php phpinfo(); ?>

- informação da versão instalada no servidor
- lista módulos extras ativos no PHP
- o PHP é configurado através do ficheiro php.ini

PHP Version 7.3.2 	
System	Windows NT LABMM 6.1 build 7601 (Windows Server 2008 R2 Standard Edition Service Pack 1) i586
Build Date	Feb 5 2019 22:51:45
Compiler	MSVC15 (Visual C++ 2017)
Architecture	x86
Configure Command	cscript /nologo configure.js "--enable-snapshot-build" "--enable-debug-pack" "--with-pdo-oci=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--with-oci8-12c=c:\php-snap-build\deps_aux\oracle\x86\instantclient_12_1\sdk,shared" "--enable-object-out-dir=../obj/" "--enable-com-dotnet=shared" "--without-analyzer" "--with-pgo"
Server API	CGI/FastCGI
Virtual Directory Support	enabled
Configuration File (php.ini) Path	C:\Windows
Loaded Configuration File	C:\Program Files (x86)\PHP\php.ini
Scan this dir for additional .ini files	(none)
Additional .ini files parsed	(none)
PHP API	20180731
PHP Extension	20180731
Zend Extension	320180731
Zend Extension Build	API320180731,TS,VC15
PHP Extension Build	API20180731,TS,VC15
Debug Build	no
Thread Safety	enabled
Thread API	Windows Threads
Zend Signal Handling	disabled
Zend Memory Manager	enabled
Zend Multibyte Support	provided by mbstring
IPv6 Support	enabled
DTrace Support	disabled
Registered PHP Streams	php, file, glob, data, http, ftp, zip, compress.zlib, compress.bzip2, https, ftps, phar
Registered Stream Socket Transports	tcp, udp, ssl, tls, tlsv1.0, tlsv1.1, tlsv1.2
Registered Stream Filters	convert.iconv.*, string.rot13, string.toupper, string.tolower, string.strip_tags, convert.*, consumed, dechunk, zlib.*, bzip2.*
This program makes use of the Zend Scripting Language Engine: Zend Engine v3.3.2, Copyright (c) 1998-2018 Zend Technologies with Zend OPcache v7.3.2, Copyright (c) 1999-2018, by Zend Technologies	
	

versão do PHP

- ▶ Como testar localmente?
 - ▶ Iniciem o XAMPP (ou outra solução que tenham)
 - ▶ Na pasta raiz da Web criem um ficheiro com extensão php (por exemplo, version.php)
 - ▶ Incluam no ficheiro o script PHP do slide anterior
 - ▶ No browser acedam a <http://localhost/version.php>
 - ▶ Devem conseguir ver uma página com informação do PHP disponível no vosso servidor local

estrutura e sintaxe

- ▶ sintaxe para PHP embebido num documento

HTML

```
<body>  
  <?php  
    echo "Hello World";  
  ?>  
</body>
```

Nota: Este código deve ser incluído na estrutura global do documento HTML!

- ▶ separação obrigatória das instruções com “;”

```
<body>  
  <?php  
    echo "Hello World";  
    echo "Peace and Love!!!!";  
  ?>  
</body>
```

estrutura e sintaxe

- ▶ podemos ter múltiplos scripts de PHP embebidos numa página HTML

```
<body>  
    <?php echo "Hello World"; ?>  
  
    <p>Outro bloco de HTML</p>  
  
    <?= "Peace and Love"; ?>  
</body>
```

Nota: Dentro de um script de PHP não podemos abrir outro script de PHP!

Dica:

```
<?= $a; ?>
```

É um atalho para

```
<?php echo $a; ?>
```

estrutura e sintaxe

- É possível ter um ficheiro só com PHP para executar operações ou, se necessário, para escrever toda uma página de HTML por script

```
<?php  
echo "Hello World";  
echo "<p>Outro pedaço de HTML</p>";  
echo "Peace and Love";
```

```
// nesta situação é recomendado não fechar o  
script de PHP
```

```
// mais tarde, vamos utilizar para criar  
funções para operações CRUD com a BD
```


comentários

```
<?php
    // Este é um comentário
    // Este é outro comentário
    echo "Hello World";

    /* Este é outro comentário
       que ocupa mais do que uma linha */
    echo "Olá Mundo";
?>
```

► comentários no estilo shell unix

```
<?php
    # Este é um comentário
    # Este é outro comentário
    echo "Hello World";
?>
```

case sensitive em PHP?

- ▶ **Case sensitive** (both user defined and PHP defined)
 - ▶ variables
 - ▶ constants
 - ▶ array keys
 - ▶ class properties
 - ▶ class constants
- ▶ **Case insensitive** (both user defined and PHP defined)
 - ▶ functions
 - ▶ class constructors
 - ▶ class methods
 - ▶ keywords and constructs (if, else, null, foreach, echo, etc...)

output

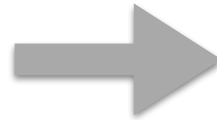
- ▶ `<?php echo "<p>Hello World</p>"; ?>`
- ▶ `<?="<p>Hello World</p>"?>`
- ▶ `<?php print "<p>Hello World</p>"; ?>`
- ▶ `<?php echo "<p>Hello World. I'm $name.</p>"; ?>`
- ▶ `<?php print "<p>Hello World. I'm $name.</p>"; ?>`

Nota: Os comandos *echo* e *print* são idênticos

'...' vs "..."

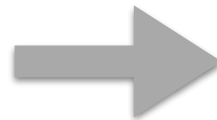
- Plicas e aspas podem produzir resultados diferentes num output em PHP

```
<?php  
    $age = 20;  
    echo "Age: $age";  
?>
```



Age: 20

```
<?php  
    $age = 20;  
    echo 'Age: $age';  
?>
```



Age: \$age

output

```
<?php printf("Bar: %d garrafas", $bottles); ?>
```

- identificador de tipo/formato da variável -> %d (inteiro)
- permite formatar a parte dinâmica da string

```
<?php $MeuTexto = sprintf("Preço: %.2f", $Price); ?>
```

- igual ao printf mas devolve o resultado para uma variável do tipo string

output

```
<?php printf("Bar: %d garrafas", $bottles); ?>
```

- Se \$bottles = 6.113 -> "**Bar: 6 garrafas**"

```
<?php $MeuTexto = sprintf("Preço: %.2f", $Price); ?>
```

- Se \$Price = 25.9183 -> \$MeuTexto = "**Preço: 25.92**"

printf e sprintf

- especificação do tipo de dados para a formatação:
 - **%b** -> número binário
 - **%c** -> caracter correspondente ao código ASCII
 - **%d** -> número inteiro com sinal
 - **%f** -> número em vírgula flutuante
 - **%o** -> número em octal
 - **%s** -> string
 - **%u** -> número inteiro sem sinal
 - **%x** -> número hexadecimal em minúsculas
 - **%X** -> número hexadecimal em maiúsculas

tipos de dados (simples)

- ▶ **boolean** (case insensitive)
 - ▶ True ($\neq 0$) ou False ($= 0$)
- ▶ **integer**
 - ▶ sem parte fracionária
 - ▶ base 16 (Hexadecimal), base 10 (Decimal), base 8 (Octal), Base 2 (Binário)
 - ▶ valor máximo = 2^{31}
- ▶ **float**
 - ▶ com parte fracionária
- ▶ **string**
 - ▶ sequência de caracteres

tipos de dados (compostos)

- ▶ Permitem agregar vários valores sob uma mesma entidade

- ▶ **array**

- ▶ `$marcacarro = array();`
 - ▶ `$marcacarro[0] = "Opel";`
 - ▶ `$marcacarro[1] = "Renault";`
 - ▶ `$marcacarro[2] = "Fiat";`

0	1	2
Opel	Renault	Fiat

- ▶ **Tipos de arrays**

- ▶ Arrays numéricos
 - ▶ Arrays associativos
 - ▶ Arrays multi-dimensionais

Nota: Vamos ter aulas dedicadas só a esta temática. Não se assustem com o slide :)

tipos de dados (compostos)

► object

- declarado através de uma classe

- ```
Class Carro {
 Propriedades -> Variáveis
 Métodos -> Funções
}
```

- instanciados através da declaração de uma variável

- ```
$CarroPolicia = new Carro();
```
- ```
$CarroLadroses = new Carro();
```

**Nota:** Não vamos ter Programação Orientada a Objetos. É abordado em LabMM5-A

# ficheiros externos em PHP

- ▶ `include "filename.php";`
- ▶ `include_once "filename.php";`
- ▶ `require "filename.php";`
- ▶ `require_once "filename.php";`

**TPC: Estudar**

[https://www.w3schools.com/php/php\\_includes.asp](https://www.w3schools.com/php/php_includes.asp)  
(muito importante para o 1º exercício prático)