Curso PHP

Do XLSX ao CMS (aula 6)

<?php echo “Hello, world!” ?>

PHP

Trata-se de uma linguagem extremamente modularizada, o que a torna ideal para instalação e uso em servidores web. Diversos módulos são criados no repositório de extensões PECL (PHP Extension Community Library).

A linguagem PHP é uma linguagem de programação de domínio específico, ou seja, seu escopo se estende a um campo de atuação que é o desenvolvimento web, embora tenha variantes como o PHP-GTK.

Seu propósito principal é de implementar soluções web velozes, simples e eficientes. (Wikipédia)

PHP: instalação

O PHP tem uma boa documentação sobre como realizar a instalação nos mais diferentes ambientes.

Disponível em: php.net/install.general.

PHP: configuração

O **php.ini** é um arquivo de configuração que contém as definições de PHP do seu servidor web.

Sempre que o PHP for iniciado, seu sistema procurará por ele e executará o arquivo para habilitar as regras de script do seu site.

Embora esteja pré-configurado, você pode precisar alterar as configurações padrões do PHP para atender às suas necessidades. (Hostinger)

PHP: built-in server

Para testar os códigos, precisamos de um ambiente de desenvolvimento.

Poderíamos utilizar ambientes com Apache ou virtualizações usando Docker. Contudo, com o intuito de simplificar e dar ênfase ao aprendizado do PHP, usaremos o servidor embutido do próprio PHP.

Esse servidor web foi desenvolvido para auxiliar no desenvolvimento de aplicações. Ele também pode ser útil para testes ou para demonstrações de aplicações que forem executadas em ambientes controlados. (PHP)

php ./bin/server

No terminal ou prompt de comando, execute o código acima

http://localhost:8000

Agora acesse a URL acima e veja as configurações do PHP

Não conseguiu?

Talvez a porta **8000** esteja em uso por outra aplicação.

Pensando nisto, há como definir um novo endereço através do comando:

- **php server localhost:8001**

Observe que a porta mudou de 8000 para 8001. Dito isto, você pode escolher qualquer uma de sua preferência.

De mesmo, *localhost* também pode ser substituído por um IP e permitir o acesso pela rede local.

Composer

Dependências e arquivos devidamente organizados

Composer

O Composer é uma ferramenta de gerenciamento de dependências em PHP.

Ele permite que você declare as bibliotecas das quais seu projeto depende e as gerenciará para você (instalação/atualização). (Hostgator)

Composer: instalação

Assim como o PHP, o Composer também possui um guia de instalação.

Disponível em: getcomposer.org/doc/00-intro.md.

composer install

No terminal ou prompt de comando, execute o comando acima

O que é vendor?

Ao executar o comando anterior, o Composer irá buscar e baixar as dependências, que irão residir no diretório **vendor**.

Você não deve modificar os códigos de um *vendor*, pois eles serão perdidos no versionamento, devido a instrução contida no .gitignore.

Autoloading

O carregamento automático de arquivos funciona de forma semelhante a instrução **include** ou **require**. (Tuts+)

Teremos funções que podem ser invocadas de qualquer parte do código. Para evitar ficar pensando o quão longe estamos da raíz do sistema e quantos níveis “precisamos subir”, o autoload faz isso para nós.

PSR

Programar também requer padrões

PHP Standard Recommendation

As recomendações criadas pelo PHP-FIG são agrupadas em PHP Standard Recommendation (PSR).

Uma PSR basicamente possui recomendações sobre um tema específico, como por exemplo, a PSR-12 que fala sobre padronização de sintaxe de código. (Treinaweb)

PSR-4

Este PSR descreve uma especificação para *autoloading classes* de caminhos de arquivo.

É totalmente interoperável e pode ser usado além de qualquer outra especificação de carregamento automático, incluindo PSR-0.

Este PSR também descreve onde colocar arquivos que serão carregados automaticamente de acordo com a especificação. (Treinaweb)

composer dump-autoload

No terminal ou prompt de comando, execute o comando acima

Dump autoload

Sempre que houver alterações no arquivo **composer.json**, é necessário que elas sejam refletidas para **vendor/autoload.php**, para isto, o comando anterior deve ser executado a cada ocorrência.

O básico do PHP

Para ser interpretado pelo PHP, o arquivo deve terminar em .php**? Nem sempre.**

```
<?php

  // ...

?>
```

Palavras reservadas

https://php.net/reserved

Palavras-chave do PHP

__halt_compiler()	abstract	and	array()	as
break	callable	case	catch	class
clone	const	continue	declare	default
die()	do	echo	else	elseif
empty()	enddeclare	endfor	endforeach	endif
endswitch	endwhile	eval()	exit()	extends
final	finally	fn (desde o PHP 7.4)	for	foreach
function	global	goto	if	implements
include	include_once	instanceof	insteadof	interface
isset()	list()	match (desde o PHP 8.0)	namespace	new
or	print	private	protected	public
readonly (desde o PHP 8.1.0) *	require	require_once	return	static
switch	throw	trait	try	unset()
use	var	while	xor	yield
yield from				

* `readonly` pode ser utilizado como nome de função

Constantes de tempo de compilação

__CLASS__	__DIR__	__FILE__	__FUNCTION__	__LINE__	__METHOD__
__NAMESPACE__	__TRAIT__				

Comentários

```php
<?php

  // comentário em linha

  # outra forma de comentário em linha

  /*
   Bloco de comentário
   Blocos podem possuir muitas linhas
   Muitas linhas
   Linhas
   .
  */
```

Variáveis

```php
<?php

  /*
   variáveis são declaradas usando $
   variáveis aceitam valores de qualquer tipo
   variáveis podem ser sobreescritas

   o ponto (.) concatenta valores
  */

  // string
  $string = 'string';

  // inteiro
  $inteiro = 1;

  // decimal
  $decimal = 1.0;

  // sobreescrito
  $string = $inteiro;

  // concatenado
  $string = 'o valor decimal é: ' . $decimal;
```

Funções

```php
<?php

  /*
   funções servem para isolar tarefas
   podem ser usada para evitar repetição de código
  */


  // declarar função
  function somar(a, b)
  {
    return a + b;
  }


  // usar função
  'A soma de 7 + 1 é: ' . somar(7, 1);
  'A soma de 7 + 2 é: ' . somar(7, 2);
  'A soma de 7 + 3 é: ' . somar(7, 3);
```

Tipagem

```php
<?php

  /*
   o PHP não é tipado, mas pode ter indução de tipos
  */


  // declarar função
  function somar(int a, int b): int
  {
    return a + b;
  }


  // usar função
  'A soma de 7 + 1 é: ' . somar(7, 1);


  // erro
  'A soma de 7 + 2.5 é: ' . somar(7, 2.5);
```

Interpolação

```php
<?php

  /*
   Valores pode ser interpolados usando aspas duplas
   O uso do delimitador '{}' não é obrigatório.
   Porém, mais a frente, será necessário usá-lo.
  */


  // declarar função
  function somar(int a, int b): int
  {
    return a + b;
  }


  // usar função
  $x = 7;
  $y = 1;

  "a soma de {$x} + {$y} é: " . somar($x, $y);
```

Condicionais

```php
<?php

  $n = 3;


  // condicional com IF/ELSE
  if ($n % 2 == 0) {
    return true;
  } else {
    return false;
  }


  // condicional com ELSE implícito
  if ($n % 2 == 0) {
    return true;
  }

  return false;
```

Mostrando em tela

```php
<?php

  echo "echo não requer parênteses.";

  // Strings podem ser passadas individualmente como múltiplos argumentos ou
  // concatenadas e passadas como um único argumento
  echo 'Esta ', 'string ', 'foi ', 'criada ', 'com múltiplos parâmetros.', "\n";
  echo 'Esta ' . 'string ' . 'foi ' . 'criada ' . 'com concatenação.' . "\n";

  // Nenhuma nova linha ou espaço são adicionados;
  // O código abaixo mostra "olámundo" em apenas uma linha
  echo "olá";
  echo "mundo";

  // O mesmo que o exemplo acima
  echo "olá", "mundo";

  echo "Esta string ocupa
múltiplas linhas. As novas linhas
também estarão na saída.";

  echo "Esta string ocupa\nmúltiplas linhas. As novas linhas\ntambém estarão na saída.";

  // O argumento pode ser qualquer expressão que produza uma string
  $foo = "exemplo";
  echo "foo é um $foo"; // foo é um exemplo
```

. PHP_EOL;