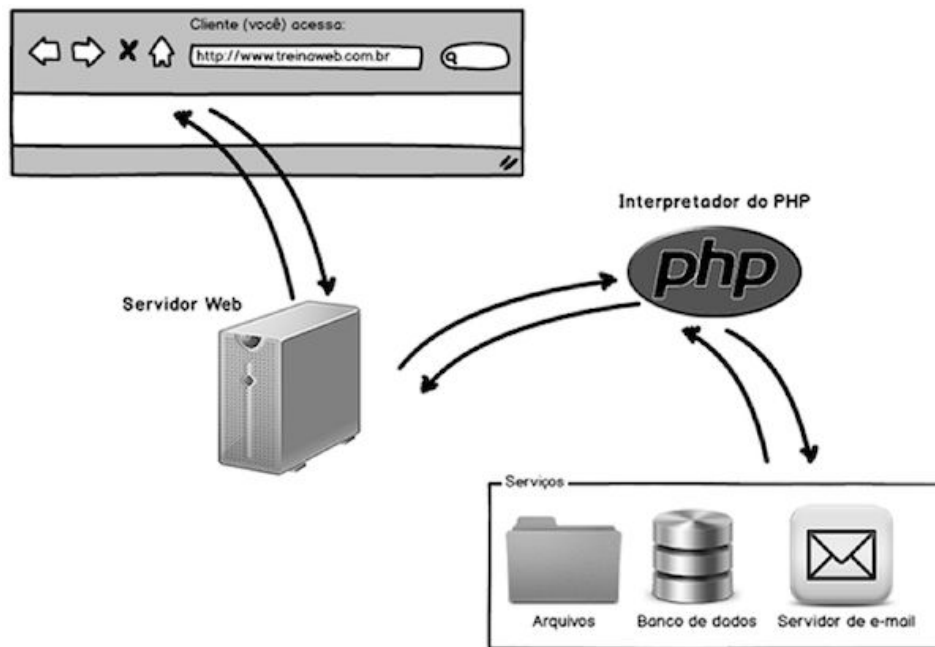


DESENVOLVIMENTO DE APLICAÇÕES

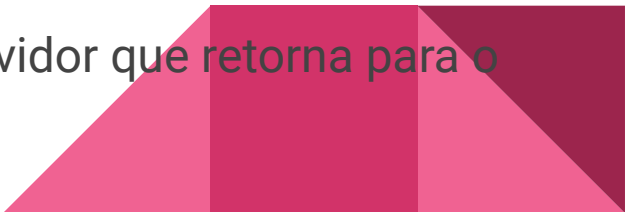
Análise e Desenvolvimento de Sistemas

Introdução ao PHP

- PHP é **server-side**, ou seja, primeiramente a página é executada/processada no servidor e então retorna um código HTML, o qual é renderizado (exibido) no navegador (cliente).



Exemplo de um passo a passo de execução:

- (1) O cliente (você) acessa o site: <http://www.unicesumar.com.br/>
 - (2) O servidor recebe essa requisição e o PHP está configurado para rodar nesse servidor. A requisição é encaminhada do servidor para o interpretador do PHP.
 - (3) O interpretador do PHP executa o script `cursos.php`.
 - (4) O script `cursos.php` possui códigos para acesso ao banco de dados que retorna a lista de cursos.
 - (5) O PHP retorna esse resultado, organizado , para o servidor que retorna para o navegador (cliente).
- 

O que pode ser feito com PHP ?

- Formulários: de contato, cadastro, e-mail e etc.
 - E-commerce: lojas virtuais.
 - Sites dinâmicos: páginas com alimentação de dados via Banco de Dados.
 - Sistemas complexos: Administração de conteúdo, gerenciamento de servidores, usuários e etc.
 - Protocolos: PHP tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir sockets e interagir com outros protocolos.
- Suporte ao padrão XML (SAX, DOM, XSLT).



Outras Vantagens

- O PHP é Multiplataforma. Aceita vários sistemas operacionais como:
 - Linux
 - Mac OS
 - Windows
- Permite a conexão direta com grande quantidade de Banco de Dados (relacionais e não-relacionais) como:
 - Oracle
 - MySQL
 - PostgreSQL
 - MongoDB
 - Etc.
- É suportado pela maioria dos servidores Web que existem no mercado:
 - APACHE
 - NGINX



Começando

- O **PHP** não é apenas uma linguagem pura. Ele é um processador/interpretador; ele procura pelas tags de abertura e fechamento, que indicam onde começar e parar de interpretar o código entre elas.
 - `"<?php"`
 - `"?>"`
- Printar informação na tela
 - `echo 'Olá mundo!';`
- Utilizar ponto e vírgula no final da instrução. `“;”`
- Extensão do arquivo é `.php`
- Sempre rodar o script em algum interpretador da linguagem.



Variáveis

Observações importantes para a declaração de variáveis em PHP:

- Os nomes de variáveis sempre começam com o caractere \$ (cifrão). Exemplo: \$idade
- São sensíveis à letra, ou seja, \$Valor é diferente de \$valor.
- Um nome de variável válido deve-se iniciar com um sublinhado OU uma letra, nunca deve ser iniciado com um número ou caractere especial.
- Seguindo as regras acima, são válidos caracteres de a-z, números e caracteres das tabelas ASCII de 128 a 255 (são alguns caracteres estendidos).
- \$this é uma variável especial da linguagem e não pode ser atribuída.



Tipagem

O PHP suporta 8 tipos de dados que podem ser divididos em três categorias:

Tipos escalares (básicos):

- Integer
- Boolean
- Float (ou "double")
- String



Tipagem

O PHP suporta 8 tipos de dados que podem ser divididos em três categorias:

Tipos compostos:

- Array
- Object



Tipagem

O PHP suporta 8 tipos de dados que podem ser divididos em três categorias:

Tipos especiais:

- Resource
- NULL



Tipagem Dinâmica

- Uma variável pode conter valores de diferentes tipos em diferentes momentos da execução do script.
- Por esse motivo não é necessário declarar o tipo de uma variável para usá-la, ou seja: não é preciso dizer ao PHP que a variável \$nome é do tipo string, pois o interpretador PHP decidirá automaticamente seu tipo, verificando o conteúdo em tempo de execução.



Arrays

Um Array pode ser usado para armazenar qualquer outro tipo de dado, incluindo outro array.

- `$meuArray = [];` ou `$meuArray = Array();`
- `$meuArray = ['valor1', 'valor2', 'valor3', ['outroArray']];`
- `$meuArray[0];` // "valor1"

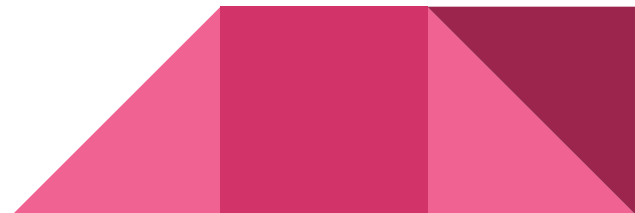


Array Associativos

Arrays associativos são definidos usando a sintaxe chave => valor. Com eles são criados contextos com significado para os Arrays.

Exemplo:

```
$alunos = ['nome' => 'João'];
```



Funções

Uma função, assim como em qualquer linguagem, é um bloco de código para facilitar tarefas repetitivas.

// exemplo

```
function soma($a, $b) {  
    return $a + $b;  
}
```



Diferenças entre POST e GET

GET

- O método GET utiliza a própria URI(url) para enviar dados ao servidor.
- O número de dados que podem ser enviados é limitado.
- GET pode ser cacheado (armazenado em cache) dependendo das configurações do navegador e também dos cabeçalhos HTTP do recurso.

O PHP fornece a variável global **`$_GET['valor']`** para entregar os valores de GET.



Diferenças entre POST e GET


POST

- O método POST envia os dados colocando-os no corpo da mensagem.
- Ou seja, ele deixa a URI separada dos dados que serão enviados (fica encapsulada). Com isso é possível enviar qualquer tipo de dado por esse método (coleções de binários como, por exemplo, imagens, música, etc., algo que não pode ser feito usando GET).

O PHP fornece a variável global **`$_POST['valor']`** para entregar os valores de POST.



Orientação a Objetos

- Um objeto representa uma coisa física, tangível, uma ideia ou conceito. Possui um estado (o que ele sabe) e um comportamento (o que ele é capaz de fazer e como reage a estímulos externos).
 - Objetos são a unidade fundamental de qualquer sistema orientado a objetos. Tudo gira em torno de objetos – eles podem representar uma pessoa, um lugar, um carro, um avião etc. As características que definem um objeto são os atributos e os comportamentos, chamados de métodos.
- 

Classes

```
1  <?php
2
3  class Carro
4  {
5      // atributos
6      public $marca;
7      public $modelo;
8      public $qtdPortas;
9      public $cor;
10     public $ano;
11
12     // métodos
13     public function acelerar()
14     {
15         echo '-> [Aumentando a aceleração do veículo ...] <br>';
16     }
17
18     public function frear()
19     {
20         echo '-> [Freando o veículo ...]';
21     }
22
23
24 }
```

Modificadores de visibilidade

- Encapsulamento é o mecanismo onde, modificadores de acesso provêm proteção aos membros internos de um objeto.
- Os modificadores de acesso definem o acesso (visibilidade) dos atributos e métodos quando a classe for instanciada ou estendida
- Uma das formas de atingir o encapsulamento é definindo a visibilidade dos métodos e propriedades.



Existem três formas de acesso:

- **public** – Quando usamos esse modificador, significa que o método ou atributo em questão pode ser acessado por todas as outras classes e métodos, sem quaisquer restrições.
- **protected** – Pode ser acessado apenas por métodos da própria classe e pelas classes-filhas.
- **private** – Modificador que não permite o acesso por classes descendentes (classes-filhas), e só pode ser acessado dentro da própria classe.



Definido Visibilidade no PHP

```
3  class Carro
4  {
5      protected $marca;
6      public $modelo;
7      private $qtdPortas;
8
9      public function setMarca($marca)
10     {
11         $this->marca = $marca;
12     }
13
14     protected function getMarca()
15     {
16         return $this->marca;
17     }
18
19     private function metodoPrivado()
20     {
21         return 'Metodo privado';
22     }
23 }
```

Métodos Construtor e Destrutor

Um “construtor” é definido pelo método: **__construct()**

- O construtor é um método especial para definir o comportamento de um objeto na hora de sua criação. Sempre que instanciamos um objeto, existindo este método, ele automaticamente será executado.

Já o método destrutor, declarado como **__destruct()**

- Ele sempre executado na destruição do objeto, ou seja, quando ele é liberado da memória.



Herança

- Em orientação a objetos, herança é o compartilhamento de atributos e comportamentos entre as classes de uma mesma hierarquia (Árvore).
- As classes inferiores da hierarquia herdam automaticamente as propriedades e os métodos das classes superiores, chamadas de Superclasses.
- Com o recurso da herança podemos criar uma classe a partir de uma já existente, herdando, reescrevendo ou acrescentando novos métodos e propriedades a ela.



Exemplo de Herança

```
3 class Funcionario extends Pessoa
4 {
5     protected $cargo;
6     protected $salario;
7
8     public function setCargo($cargo)
9     {
10         $this->cargo = ucfirst($cargo);
11     }
12
13     public function getCargo()
14     {
15         return $this->cargo;
16     }
17
18     public function setSalario($salario)
19     {
20         $this->salario = $salario;
21     }
22
23     public function getSalario()
24     {
25         return number_format($this->salario, 2, ',', '.');
26     }
27 }
```

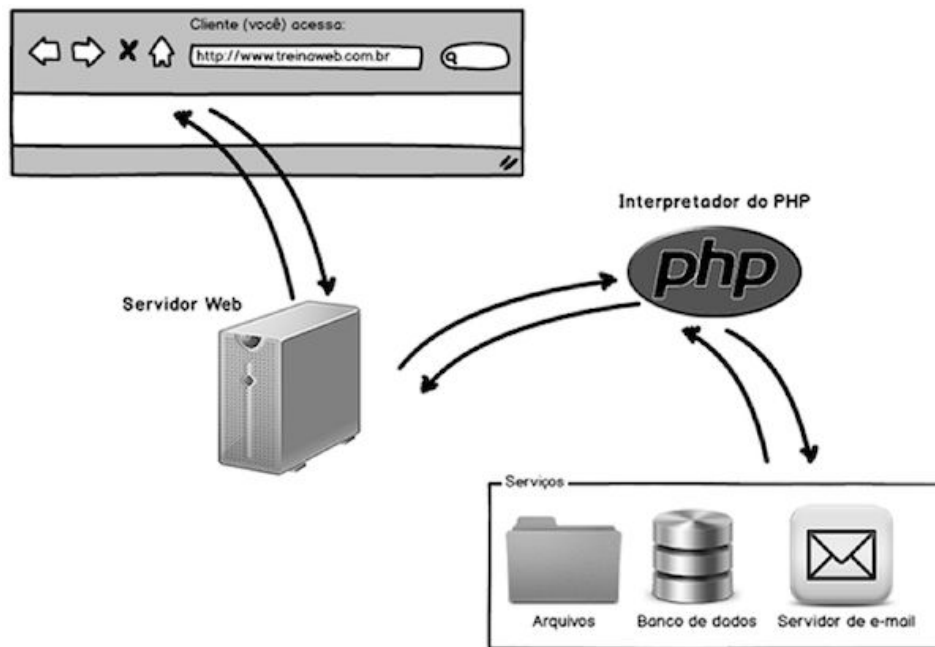

Última Aula

- Preparação do ambiente de desenvolvimento PHP
 - Instalar o VirtualBox;
 - Instalar o Vagrant;
 - Atualizar o PS > 3.0;
 - Clonar o Homestead;
 - Baixar a máquina virtual;
 - Configurar o HOST e o Homestead.YAML
 - Testar no navegador o endereço respondendo.

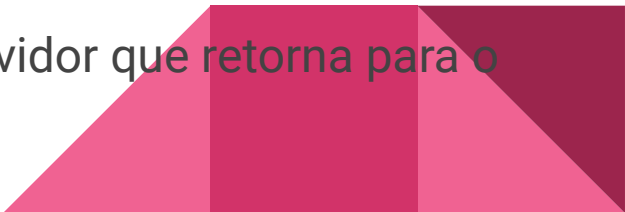


Introdução ao PHP

- PHP é **server-side**, ou seja, primeiramente a página é executada/processada no servidor e então retorna um código HTML, o qual é renderizado (exibido) no navegador (cliente).



Exemplo de um passo a passo de execução:

- (1) O cliente (você) acessa o site: <http://www.unicesumar.com.br/>
 - (2) O servidor recebe essa requisição e o PHP está configurado para rodar nesse servidor. A requisição é encaminhada do servidor para o interpretador do PHP.
 - (3) O interpretador do PHP executa o script `cursos.php`.
 - (4) O script `cursos.php` possui códigos para acesso ao banco de dados que retorna a lista de cursos.
 - (5) O PHP retorna esse resultado, organizado , para o servidor que retorna para o navegador (cliente).
- 

O que pode ser feito com PHP ?

- Formulários: de contato, cadastro, e-mail e etc.
 - E-commerce: lojas virtuais.
 - Sites dinâmicos: páginas com alimentação de dados via Banco de Dados.
 - Sistemas complexos: Administração de conteúdo, gerenciamento de servidores, usuários e etc.
 - Protocolos: PHP tem suporte a outros serviços através de protocolos como IMAP, SNMP, NNTP, POP3 e, logicamente, HTTP. Ainda é possível abrir sockets e interagir com outros protocolos.
- Suporte ao padrão XML (SAX, DOM, XSLT).



Outras Vantagens

- O PHP é Multiplataforma. Aceita vários sistemas operacionais como:
 - Linux
 - Mac OS
 - Windows
- Permite a conexão direta com grande quantidade de Banco de Dados (relacionais e não-relacionais) como:
 - Oracle
 - MySQL
 - PostgreSQL
 - MongoDB
 - Etc.
- É suportado pela maioria dos servidores Web que existem no mercado:
 - APACHE
 - NGINX



Começando

- O **PHP** não é apenas uma linguagem pura. Ele é um processador/interpretador; ele procura pelas tags de abertura e fechamento, que indicam onde começar e parar de interpretar o código entre elas.
 - `"<?php"`
 - `"?>"`
- Printar informação na tela
 - `echo 'Olá mundo!';`
- Utilizar ponto e vírgula no final da instrução. `“;”`
- Extensão do arquivo é `.php`
- Sempre rodar o script em algum interpretador da linguagem.



Variáveis

Observações importantes para a declaração de variáveis em PHP:

- Os nomes de variáveis sempre começam com o caractere \$ (cifrão). Exemplo: \$idade
- São sensíveis à letra, ou seja, \$Valor é diferente de \$valor.
- Um nome de variável válido deve-se iniciar com um sublinhado OU uma letra, nunca deve ser iniciado com um número ou caractere especial.
- Seguindo as regras acima, são válidos caracteres de a-z, números e caracteres das tabelas ASCII de 128 a 255 (são alguns caracteres estendidos).
- \$this é uma variável especial da linguagem e não pode ser atribuída.



Tipagem

O PHP suporta 8 tipos de dados que podem ser divididos em três categorias:

Tipos escalares (básicos):

- Integer
- Boolean
- Float (ou "double")
- String



Tipagem

O PHP suporta 8 tipos de dados que podem ser divididos em três categorias:

Tipos compostos:

- Array
- Object



Tipagem

O PHP suporta 8 tipos de dados que podem ser divididos em três categorias:

Tipos especiais:

- Resource
- NULL



Tipagem Dinâmica

- Uma variável pode conter valores de diferentes tipos em diferentes momentos da execução do script.
- Por esse motivo não é necessário declarar o tipo de uma variável para usá-la, ou seja: não é preciso dizer ao PHP que a variável \$nome é do tipo string, pois o interpretador PHP decidirá automaticamente seu tipo, verificando o conteúdo em tempo de execução.



Arrays

Um Array pode ser usado para armazenar qualquer outro tipo de dado, incluindo outro array.

- `$meuArray = [];` ou `$meuArray = Array();`
- `$meuArray = ['valor1', 'valor2', 'valor3', ['outroArray']];`
- `$meuArray[0];` // "valor1"



Array Associativos

Arrays associativos são definidos usando a sintaxe chave => valor. Com eles são criados contextos com significado para os Arrays.

Exemplo:

```
$alunos = ['nome' => 'João'];
```



Funções

Uma função, assim como em qualquer linguagem, é um bloco de código para facilitar tarefas repetitivas.

// exemplo

```
function soma($a, $b) {  
    return $a + $b;  
}
```



Diferenças entre POST e GET

GET

- O método GET utiliza a própria URI(url) para enviar dados ao servidor.
- O número de dados que podem ser enviados é limitado.
- GET pode ser cacheado (armazenado em cache) dependendo das configurações do navegador e também dos cabeçalhos HTTP do recurso.

O PHP fornece a variável global **`$_GET['valor']`** para entregar os valores de GET.



Diferenças entre POST e GET

POST

- O método POST envia os dados colocando-os no corpo da mensagem.
- Ou seja, ele deixa a URI separada dos dados que serão enviados (fica encapsulada). Com isso é possível enviar qualquer tipo de dado por esse método (coleções de binários como, por exemplo, imagens, música, etc., algo que não pode ser feito usando GET).

O PHP fornece a variável global **`$_POST['valor']`** para entregar os valores de POST.



Orientação a Objetos

- Um objeto representa uma coisa física, tangível, uma ideia ou conceito. Possui um estado (o que ele sabe) e um comportamento (o que ele é capaz de fazer e como reage a estímulos externos).
- Objetos são a unidade fundamental de qualquer sistema orientado a objetos. Tudo gira em torno de objetos – eles podem representar uma pessoa, um lugar, um carro, um avião etc. As características que definem um objeto são os atributos e os comportamentos, chamados de métodos.



Classes

```
1  <?php
2
3  class Carro
4  {
5      // atributos
6      public $marca;
7      public $modelo;
8      public $qtdPortas;
9      public $cor;
10     public $ano;
11
12     // métodos
13     public function acelerar()
14     {
15         echo '-> [Aumentando a aceleração do veículo ...] <br>';
16     }
17
18     public function frear()
19     {
20         echo '-> [Freando o veículo ...]';
21     }
22
23
24 }
```

Modificadores de visibilidade

- Encapsulamento é o mecanismo onde, modificadores de acesso provêm proteção aos membros internos de um objeto.
- Os modificadores de acesso definem o acesso (visibilidade) dos atributos e métodos quando a classe for instanciada ou estendida
- Uma das formas de atingir o encapsulamento é definindo a visibilidade dos métodos e propriedades.



Existem três formas de acesso:

- **public** – Quando usamos esse modificador, significa que o método ou atributo em questão pode ser acessado por todas as outras classes e métodos, sem quaisquer restrições.
- **protected** – Pode ser acessado apenas por métodos da própria classe e pelas classes-filhas.
- **private** – Modificador que não permite o acesso por classes descendentes (classes-filhas), e só pode ser acessado dentro da própria classe.



Definido Visibilidade no PHP

```
3  class Carro
4  {
5      protected $marca;
6      public $modelo;
7      private $qtdPortas;
8
9      public function setMarca($marca)
10     {
11         $this->marca = $marca;
12     }
13
14     protected function getMarca()
15     {
16         return $this->marca;
17     }
18
19     private function metodoPrivado()
20     {
21         return 'Metodo privado';
22     }
23 }
```

Métodos Construtor e Destrutor

Um “construtor” é definido pelo método: **__construct()**

- O construtor é um método especial para definir o comportamento de um objeto na hora de sua criação. Sempre que instanciamos um objeto, existindo este método, ele automaticamente será executado.

Já o método destrutor, declarado como **__destruct()**

- Ele sempre executado na destruição do objeto, ou seja, quando ele é liberado da memória.



Herança

- Em orientação a objetos, herança é o compartilhamento de atributos e comportamentos entre as classes de uma mesma hierarquia (Árvore).
- As classes inferiores da hierarquia herdam automaticamente as propriedades e os métodos das classes superiores, chamadas de Superclasses.
- Com o recurso da herança podemos criar uma classe a partir de uma já existente, herdando, reescrevendo ou acrescentando novos métodos e propriedades a ela.



Exemplo de Herança

```
3 class Funcionario extends Pessoa
4 {
5     protected $cargo;
6     protected $salario;
7
8     public function setCargo($cargo)
9     {
10         $this->cargo = ucfirst($cargo);
11     }
12
13     public function getCargo()
14     {
15         return $this->cargo;
16     }
17
18     public function setSalario($salario)
19     {
20         $this->salario = $salario;
21     }
22
23     public function getSalario()
24     {
25         return number_format($this->salario, 2, ',', '.');
26     }
27 }
```