

Aula 5

Professor:

Inês Dutra

Funções, conversão de tipos,
programação orientada a objetos

PHP: Funções

- Úteis para definir partes de código que se repetem
- Por exemplo, suponha que o programa precise escrever o logo de uma empresa em vários lugares do código
- O código referente ao logo pode ser codificado apenas uma vez

PHP: Funções

- Exemplo

```
function display_logo()
{
    echo '<hr width="50" align="left">', "\n";
    echo '<br>', "\n";
    echo '<hr width="50" align="left"><br>',
          "\n";
    return;
}
```

PHP: Funções

- Vantagens da utilização de funções (mesmas de todas as linguagens que provêem funções)
 - ▶ Menos digitação
 - ▶ Código mais compacto
 - ▶ Mais fácil de ler
 - ▶ Mais fácil de manter

PHP: Funções

- Escopo das variáveis em funções
 - ▶ Variáveis criadas dentro da função são locais, ou seja, não existem fora da função
 - ▶ Podem ser vistas fora da função utilizando-se o comando `global`

PHP: Funções

```
function format_nome()  
{  
    $prim_nome = "Maria";  
    $sobrenome = "Silva";  
    $nome = $sobrenome.", ". $prim_nome;  
}  
format_nome();  
echo "$nome"; ← Nada é impresso!!
```

PHP: Funções

```
function format_nome()
{
    $prim_nome = "Maria";
    $sobrenome = "Silva";
    global $nome; ←
    $nome = $sobrenome.", ". $prim_nome;
}

format_nome();
echo "$nome"; ← Silva, Maria é impresso!
```

PHP: Funções

- Assim como variáveis de funções não são vistas pelo programa, variáveis que são criadas no programa, fora da função, também não são vistas pela função
- Se, na função, queremos ter acesso aos valores de variáveis que estão fora dela, deve-se também utilizar o comando

PHP: Funções

```
$prim_nome = "Maria";
$sobrenome = "Silva";
function format_nome()
{
    global $prim_nome, $sobrenome;
    $nome = $sobrenome.", ". $prim_nome;
}
format_nome();
```

- Neste exemplo, **\$prim_nome** e **\$sobrenome** dentro da função são as mesmas variáveis do programa

PHP: Funções, passagem de parâmetros

- Parâmetros podem ser passados na chamada de uma função
- Arrays podem ser passados como parâmetro
- Se faltar algum parâmetro na chamada, a função assume valor:
 - ▶ String vazia para variáveis do tipo string
 - ▶ Zero(0) para números
- A função pode também assumir valores default para os parâmetros
- Passagem de parâmetros pode ser por **valor** ou por **referência**

PHP: Funções, passagem de parâmetros

```
function calcula_soma($n1, $n2)
{
    return $n1 + $n2;
}

calcula_soma(20, 30);      -----> 50
calcula_soma(20*20, 32+2); -----> 434
calcula_soma();           -----> 0!!
```

[Abrir editor](#)[Abrir navegador](#)

PHP: Funções, passagem de parâmetros

```
function calcula_soma($n1=1, $n2=1)
{
    return $n1 + $n2;
}
calcula_soma(20, 30);      -----> 50
calcula_soma(20*20, 32+2); -----> 434
calcula_soma();           -----> 2!!
```

[Abrir editor](#)[Abrir navegador](#)

PHP: Funções, passagem de parâmetros

```
$numeros = array(10,20,30,40,50);
soma($numeros);
function soma($array)
{
    for ($i=0; $i<sizeof($array); $i++)
    {
        $soma += $array[$i];
    }
    echo $soma; → 150
}
```

[Abrir editor](#)[Abrir navegador](#)

PHP: Funções, passagem de parâmetros por referência

```
function dobro(&$var)
{
    $var = $var * 2;
}
$var1 = 5;
dobra($var1);
echo $var1; → 10
```

[Abrir editor](#)[Abrir navegador](#)

PHP: Uso de referências

- Referências podem também ser usadas em atribuições
- Exemplo

```
$x = 5;  
$y = &$x;  
$x ++;  
echo $x; → 6!  
echo $y; → 6!
```

PHP: Funções

- Funções (ou qualquer código PHP) podem ser guardadas em arquivos separados
- São incluídas no programa principal com os comandos:
 - ▶ `include`
 - ▶ `require`

PHP: Funções

- Exemplo:

Arquivo **soma.inc**

```
<?php  
function soma($n1, $n2)  
{  
    return $n1+$n2;  
}  
?>
```

PHP: Funções

- Programa:

```
<html>
<head>
    <title> Chamada de função </title>
</head>
<body bgcolor="#ffffff">
<?
    include "soma.inc";
    $result = soma(20,34);
    echo $result;
?>
</body>
</html>
```

[Abrir editor](#)[Abrir navegador](#)

PHP: `include` X `require`

- `require` é executado antes da execução do script, obrigando que o arquivo seja sempre inserido no corpo do programa
- `include` é interpretado junto com a execução do script

PHP: Conversão de tipos

- Pode ser automática ou explícita
- Funções para conversão explícita de tipos:
 - ▶ `string strval(mixed var)`
 - ▶ `integer intval(mixed var)`
 - ▶ `float floatval(mixed var)`

PHP: Conversão de tipos

- A função **settype(mixed var, type)** muda o tipo da variável **var** para o tipo escolhido **type**, que pode ser:
 - ▶ **array**
 - ▶ **boolean**
 - ▶ **float**
 - ▶ **integer**
 - ▶ **string**
- Exemplo:

```
$var = "4tro"; settype($var, "integer");
```

[Abrir editor](#)[Abrir navegador](#)

PHP: Conversão de tipos

<code>\$var=</code>	(int) \$var	(bool) \$var	(string) \$var	(float) \$var
<code>null</code>	0	false	<code>""</code>	0
<code>true</code>	1	true	<code>"1"</code>	1
<code>false</code>	0	false	<code>""</code>	0
<code>0</code>	0	false	<code>"0"</code>	0
<code>3.8</code>	3	true	<code>"3.8"</code>	3.8
<code>"0"</code>	0	false	<code>"0"</code>	0
<code>"10"</code>	10	true	<code>"10"</code>	10
<code>"6 m"</code>	6	true	<code>"6 m"</code>	6
<code>"foo"</code>	0	true	<code>"foo"</code>	0

PHP: Conversão automática de tipos

```
$var = "100" + 15;           → 115  
$var = "100" + 15.0;         → 115.0  
$var = 39. " passos";       → 39 passos  
$var = 39 + " passos";      → 39  
$var = 40 + "1 ladrão";     → 41
```

PHP: verificação de tipos, funções úteis

- `is_int`
- `is_float`
- `is_array`
- `is_string`
- `is_bool`
- `is_object`
- `var_dump`
- `isset`
- `empty`

PHP: Programação orientada a objetos

- PHP implementa uma forma limitada de programação orientada a objetos, onde o programador pode definir **classes** e criar **objetos** que são instâncias destas classes
- Uma classe define uma estrutura de dados composta por variáveis e métodos

PHP: objetos, exemplo

```
<?
class contador;
{
    // variáveis
    var $cont = 0;
    var $início = 0;
    // Métodos
    function IniciaContadorEm($i)
    {
        $this->cont = $i;
        $this->início = $i;
    }
    function Incrementa()
    {
        $this->cont++;
    }
    function reset()
    {
        $this->cont = $this->início;
    }
    function EscreveValor()
    {
        print $this->cont;
    }
}
?>
```

PHP: objetos

- Para utilizar as estruturas de dados e funções definidas em uma classe, devemos criar uma **instância** desta classe
- A instância é um **objeto**
- Uma nova instância (objeto) é criada com o comando **new**

PHP: objetos

```
<html>
<head>
<title> Exemplo de utilização de classes e objetos </title></head>
<body>
<h1> Exemplo de utilização de classes e objetos </h1>
<?
include "cont.inc";
$Ocont = new contador;
// começa contador em 10
$Ocont->IniciaContadorEm(10);
// incrementa o contador
$Ocont->incrementa();
$Ocont->incrementa();
$Ocont->incrementa();
// Escreve valor atual do contador
echo "<p>Valor atual do contador: ";
$Ocont->EscreveValor(); → 13!
$Ocont->reset();
echo "<p>Valor atual do contador: ";
$Ocont->EscreveValor(); → 10!
```

[Abrir editor](#)[Abrir navegador](#)

PHP: classes e herança

- Conceito poderoso em programação orientada a objetos
- Permite que novas classes sejam criadas a partir de classes já criadas
- A nova classe passa a ser uma extensão da classe original

PHP: herança, exemplo

- Suponha que queremos escrever uma nova classe que utilize a classe Contador como base para criar objetos do tipo caixa de garrafas, onde cada caixa tenha 12 garrafas.

PHP: herança, exemplo

```

<?
    include "cont.inc";
    class ContGarrafas extends Contador
    {
        // Adiciona 12 garrafas ao contador
        function AdicionaCaixa()
        {
            $this->cont += 12;
        }
        function NúmerodeCaixas()
        {
            return ceil($this->cont/12);
        }
        function ContGarrafas($início) → construtor
        {
            $this->cont = $início;
        }
    }
    $temp = new ContGarrafas(12); → $cont = 12
    $temp->AdicionaCaixa(); → $cont = 24
    $var = $temp->NúmerodeCaixas(); → $var = 2
?>

```

[Abrir editor](#)
[Abrir navegador](#)