Engenharia de Telecomunicações e Rede de Computadores

Desenvolvimento Web

Aula III

VISÃO GERAL DO PHP

O que é PHP?

- É uma linguagem de criação de scripts do lado do servidor que foi projetada especificamente para a Web.
- Dentro de uma página HTML podemos embutir código PHP que será executado toda vez que a página for visitada.
- O código PHP interpretado no servidor Web gera como saída para o visitante uma página HTML.

O que é PHP?

- O PHP foi concebido em 1994 por *Rasmus Lerdorf*.
- É um produto *Open Source*, o que significa que temos acesso ao código-fonte. É possível utilizá-lo, alterá-lo e redistribuí-lo sem pagar nada.
- Originalmente significava *Personal Home Page*, mas foi alterado de acordo com a convenção recursiva do GNU para *PHP Hypertext Preprocessor*.

Quem utiliza o PHP?

Principais Características

Desempenho

- Utilizando um único servidor sem grandes investimentos, é possível atender milhões de acessos por dia.

Principais Características

Integração com banco de dados

- Existem conexões nativas disponíveis para os principais banco de dados do mercado. Além do *MySQL*, é possível conectar-se diretamente com *PostgreSQL*, *Oracle*, *MS SQL Server InterBase*, *Sybase*, entre outros.
- Utilizando o *Open Data Base Connectivity* (ODBC) é possível realizar conexão com qualquer banco de dados que forneça um driver *ODBC*.

Principais Características

Bibliotecas integradas

- Como o PHP foi projetado para utilização na Web, ele possui diversas funções integradas para realizar tarefas úteis relacionadas à Web.
- É possível se conectar a outros serviços de rede, enviar e-mails, trabalhar com cookies e gerar documentos PDF com algumas linhas de código.

Principais Características

Custo

- O PHP é gratuito, você pode fazer o download da última versão a qualquer momento em http://www.php.net.

Principais Características

Aprendizado

- A sintaxe do PHP é baseada em outras linguagens tradicionais de programação, principalmente C e Perl.

Principais Características

Portabilidade

- É possível executar código PHP em sistemas operacionais do tipo *Unix* gratuitos como o *Linux* e o *FreeBSD*, versões comerciais do *Unix* como *Solaris* e *IRIX*, ou em versões diferentes do *Microsoft Windows*.

REQUSITITOS PARA UTILIZAÇÃO DO PHP

Requisitos

Servidor	Sistema Operacional
Apache Web Server	Linux, MacOS e Windows
Internet Information Services	Windows

UTILIZANDO O PHP

Utilizando o PHP

- Uma da aplicações mais comuns de qualquer linguagem de criação de scripts do lado do servidor é processar formulários de HTML.
- Iniciaremos o aprendizado do PHP com a manipulação de formulários.

Utilizando o PHP

Formulário de pedido

- Criaremos um formulário de pedido de produtos. Este formulário de pedido é relativamente simples, semelhante a muitos que encontramos na Web.
- Apresentaremos o que o cliente encomendou, o total do pedido e o valor do imposto sobre a venda a ser acrescido ao pedido.

Utilizando o PHP

Item		Quantidade
	Macaco hidráulico R$60,50	
	Aditivo para radiador R$17,10	
	DVD automotivo R$629,00	
Enviar Pedido		

Utilizando o PHP

Formulário de pedido

- A ação do formulário será configurada para um script PHP que processará o pedido do cliente.

```
<form method="post" action="processarPedido.php">
```

Utilizando o PHP

Formulário de pedido

- Foram atribuídos nomes aos campos de texto do formulário. Os mesmos serão recuperados no script de PHP.

```
<input name="macaco_qtd" type="text" />

<input name="aditivo_qtd" type="text" />

<input name="dvd_qtd" type="text" />
```

Utilizando o PHP

Processando o Formulário

- Para processarmos o formulário precisaremos criar o script PHP mencionado no atributo *ACTION* da tag *FORM*.
- Criaremos um arquivo PHP chamado **processarPedido.php**.

Utilizando o PHP

Embutindo o PHP no HTML

```
<body>
<?php
    echo '<p>Pedido realizado com sucesso.</p>';
?>
</body>
```

- Note que o código PHP foi embutido dentro do HTML.

Utilizando o PHP

- Visualizando o código fonte no navegador de Internet temos a seguinte saída:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>Desenvolvimento Web</title>
</head>

<body>

<p>Pedido realizado com sucesso.</p></body>
</html>
```

- Nada do PHP bruto é visível, o PHP foi interpretado e executado do lado do servidor Web.

Utilizando tags de PHP

- O código de PHP no exemplo anterior iniciava com **<?php** e terminava com **?>**.
- Esses símbolos são chamados de tags de PHP que dizem ao servidor Web onde o código de PHP inicia e onde termina.
- Qualquer texto fora dessas tags será tratado como HTML normal. As tags de PHP permitem *escapar* da HTML.

Estilo de tags do PHP

- Existem quatro estilos diferente de tags de PHP que podem ser utilizados.
  - Estilo XML `<?php ?>`
  - Estilo abreviado `<? ?>`
  - Estilo SCRIPT `<script language="php"> </script>`
  - Estilo ASP `<% %>`

Instruções PHP

- Dizemos ao interpretador do PHP o que fazer inserindo as instruções do PHP entre as tags de abertura e fechamento.Até o momento utilizamos somente um tipo de instrução:

```
<?php
    echo '<p>Pedido realizado com sucesso.</p>';
?>
```

A instrução **echo** imprime uma determinada string passada como parâmetro.

Um ponto e vírgula deve ser colocado ao final de cada instrução PHP.

Comentários

- Os comentários no código fonte atuam como notas para as pessoas que leem o código.
  - Comentários de diversas linhas devem iniciar com **/*** e terminar com ***/**.
  - Comentários de uma única linha devem vir após **//**.

Adicionando conteúdo dinâmico

- A principal razão de utilizar uma linguagem de criação de script do lado servidor é ser capaz de fornecer **conteúdo dinâmico** para usuários de um site.
- Essa é uma aplicação importante porque isso muda o conteúdo de acordo com a necessidade de um usuário ou ao longo tempo manterá os visitantes voltando para um site.

O PHP permite fazer isso facilmente.

Adicionando conteúdo dinâmico

- Vamos iniciar com um exemplo simples:

```
<?php
    echo '<p>Pedido realizado com sucesso.</p>';
    echo date('h:i:s - j/m/y');
?>
```

- Neste código estamos utilizando a função *date()* predefinida do PHP para dizer ao cliente a data e hora em que o pedido foi processado.

Para maiores detalhes sobre a função *date()* acesse:
http://php.net/manual/pt_BR/function.date.php

Acessando variáveis do formulário

- Dentro do script do PHP, podemos acessar cada um dos campos de formulário como uma variável do PHP cujo nome referencia o nome do campo do formulário.
- Podemos acessar o conteúdo do campo *aditivo_qtd* da seguinte maneira:

```
$_POST['aditivo_qtd'];
```

Acessando variáveis do formulário


```
<input name="aditivo_qtd" type="text" />
```


```
$_POST['aditivo_qtd'];
```

Acessando variáveis do formulário

```php
<?php
    echo '<p>Pedido realizado com sucesso no dia ';
    echo date('d/m/y - h:i:s');
    echo '</p>';
    echo '<p>Resumo do pedido:</p>';

    $aditivo_qtd = $_POST['aditivo_qtd'];
    $macaco_qtd  = $_POST['macaco_qtd'];
    $dvd_qtd     = $_POST['dvd_qtd'];

    echo $aditivo_qtd . ' aditivo(s). </br>';
    echo $macaco_qtd . ' macacos hidraulico(s). </br>';
    echo $dvd_qtd . ' dvd(s).';
?>
```

Concatenação de string

- No script, utilizamos *echo* para imprimir o valor que o usuário digitou em cada um dos campos de formulário, seguido por algum texto explanatório.

```
echo $_POST['aditivo_qtd'] . ' aditivo(s). </br>';
```

Operador de concatenação de string utilizado para adicionar partes de texto.

Variáveis e literais

- **Variáveis** são símbolos para dados.
- As **strings** são dados próprios. Quando utilizamos partes de dados brutos em um programa, o chamamos de **literal** para distingui-lo de uma variável.
- *$aditivo_qtd* é uma variável, um símbolo que representa os dados que o cliente digitou.
- Por outro lado, *' aditivo(s). </br>'* é um literal. Ele pode ser considerado pelo ser valor nominal.

Identificadores

- Os identificadores são os nomes de variáveis. Há algumas regras simples sobre identificadores:
  - Os identificadores podem ser de qualquer comprimento e podem consistir em letras, números, sublinhados e sinais de cifrão. Mas você deve ter cuidado ao utilizar sinais de cifrão em identificadores.
  - Os identificadores não podem iniciar com um dígito.
- NO PHP, os identificadores diferenciam letras maiúsculas de minúsculas.

Atribuindo valores a variáveis

- Podemos atribuir valores a variáveis utilizando o operador de atribuição, =, como fizemos ao copiar um valor de variável para outra.

```
$aditivo_qtd = $_POST['aditivo_qtd'];
```

Tipos de variável

- O tipo de uma variável refere-se ao tipo de dados que é armazenado nela. O PHP suporta os seguintes tipos de dados:
  - Integer (Inteiro) – Utilizado para números inteiros
  - Double (Dupla precisão) – Utilizado para números reais
  - String – Utilizado para strings de caracteres
  - Booleano – Utilizado para valores verdadeiros ou falsos
  - Array – Utilizado para armazenar vários itens de dados do mesmo tipo
  - Objeto – Utilizado para armazenar instância de classes

Força do Tipo

- O PHP é uma linguagem de tipificação fraca.
- Na maioria das linguagens de programação as variáveis só podem armazenas um tipo de dados e este tipo deve ser declarado antes da variável ser utilizada.
- No PHP, o tipo de uma variável é determinado pelo valor atribuído a ela.

Força do Tipo

```php
$total = 0;
```

Como atribuímos 0, um inteiro, agora esta variável é do tipo **Inteiro**.

```php
$total = 'Teste';
```

Como atribuímos um texto, um literal, agora esta variável é do tipo **String**.

O PHP altera o tipo da variável a qualquer momento de acordo com o que está armazenado nela.

Constantes

- Vimos anteriormente que podemos alterar o valor armazenado em uma variável. Também podemos declarar constantes.
- Uma constante armazena um valor como uma variável, mas seu valor é configurado uma vez e então não pode ser alterado em outra parte no script.

Escopo de Variável

- O termo escopo refere-se aos lugares dentro de um script nos quais uma variável é visível.
  - As **variáveis super globais** são visíveis por toda parte dentro de um script.
  - As **variáveis globais** declaradas em um script são visíveis por todo este script, mas não dentro de funções.
  - As **variáveis locais** são utilizadas e visíveis dentro das funções.

Operadores

- Os operadores são símbolos que podem ser utilizados para manipular valores e variáveis realizando uma operação neles. Precisaremos utilizar algum desses operadores para elaborar os totais e o imposto sobre o pedido do cliente.

Operadores Aritméticos

- Os operadores aritméticos são muito simples e diretos – eles são simplesmente os operadores matemáticos normais.

Operador	Nome	Exemplo
+	Adição	$a + $b
-	Subtração	$a - $b
*	Multiplicação	$a * $b
/	Divisão	$a / $b
%	Módulo	$a % $b

Operadores de String

- Podemos utilizar o operador de concatenação de string para juntar duas strings e gerar e armazenar um resultado da mesma maneira como utilizaria o operador de adição para somar dois números.

```
$a = 'Desenvolvimento';
$b = 'Web';
$resultado = $a.$b;
```

Operadores de Atribuição

- Vimos anteriormente que **=** é o operador de atribuição básico. Sempre se refira a esse como operador de atribuição e o leia como “é configurado ou definido como”.

```
$total = 0;
```

Operadores de Atribuição de Combinação

- Além da atribuição simples, há um conjunto de operadores de atribuição combinados. Cada um desses operadores é um modo abreviado de fazer outra operação com uma variável e atribuir o resultado de volta a essa variável. Por exemplo:

```
$a += 5;
```
é equivalente a escrever:
```
$a = $a + 5;
```

Operadores de Atribuição de Combinação

Operador	Utilização	Equivalente a
+=	$a += $b	$a= $a + $b
-=	$a - $b	$a= $a - $b
*=	$a *= $b	$a= $a * $b
/=	$a /= $b	$a= $a / $b
%=	$a %= $b	$a= $a % $b

Pré- e pós-incremento e decremento

- Os operadores de pré e pós-incremento (++) e decremento (--) são semelhantes aos operadores += e _=, mas com duas variações.
- Todos os operadores de incremento têm dois efeitos: eles incrementam e atribuem um valor.

```
$a = 4;
echo ++$a . '</br>';
echo $a . '</br>';
```

```
5
5
```

```
$b = 4;
echo $b++ . '</br>';
echo $b;
```

```
4
5
```

Operadores de Comparação

- Os operadores de comparação são utilizados para comparar dois valores. As expressões que utilizam esses operadores retornam um valor lógico, *true* ou *false*, dependendo do resultado da comparação.

Operador de Igualdade

- Operador de comparação de igualdade, **==** (dois sinais de igual) permite testar se dois valores são iguais. Por exemplo, poderíamos utilizar a expressão:

```
$a == $b
```

- Para testar se os valores armazenados em $a e $b são os mesmos. O resultado retornado por essa expressão será *true* se eles forem iguais ou *false* se não forem iguais.

Outros Operadores de Comparação

- O PHP também suporta outros operadores de comparação.

Operador	Nome	Utilização
==	igual a	$a == $b
===	idêntico a	$a === $b
!=	não igual	$a != $b
< >	não igual	$a < > $b
<	menor que	$a < $b
>	maior que	$a > $b
<=	menor ou igual a	$a <= $b
>=	maior ou igual a	$a >= $B

Operadores Lógicos

- Os operadores lógicos são utilizados para combinar os resultados de condições logicas. Por exemplo, poderíamos estar interessados em um caso em que o valor de uma variável, $a, está estre 0 e 100. Precisaríamos testar as condições $a >= 0 e $a <= 100, utilizando o operador AND, assim:

```
$a >= 0 && $a <= 100
```

Operadores Lógicos

Operador	Nome	Utilização	Resultado
!	NOT	!$b	Retorna true se $b for false e vice-versa
&&	AND	$a && $b	Retorna true se $a e $b forem true; caso contrário, false
II	OR	$a II $b	Retorna true se $a ou $b ou ambos forem true; caso contrário, false
And	AND	$a and $b	O mesmo que &&, mas com precedência mais baixa
Or	OR	$a or $b	O mesmo que II, mas com precedência mais baixa

UTILIZANDO OPERADORES: CALCULANDO OS TOTAIS DO FORMULÁRIO

Calculando os Totais do Formulário

- Agora que sabemos utilizar os operadores do PHP, podemos calcular os totais e o imposto no formulário de pedido.
- Adicione o código a seguir após o código utilizado para acessar as variáveis dos formulário.

Calculando os Totais do Formulário

```php
$total_qtd  = 0;

$total_qtd = $aditivo_qtd + $macaco_qtd + $dvd_qtd;
echo 'Total de itens do pedido:  '.$total_qtd.'</br>';

$total_soma = 0.00;

define ('aditivo_preco', 17.10);
define ('macaco_preco', 60.50);
define ('dvd_preco', 629);

$total_soma =  $aditivo_qtd * aditivo_preco
             + $macaco_qtd  * macaco_preco
             + $dvd_qtd     * dvd_preco;

echo 'Subtotal do pedido: R$'.number_format($total_soma, 2,',','.').'</br>';

$imposto = 0.10;
$total_soma = $total_soma * (1 + $imposto);
echo 'Total com impostos: R$'.number_format($total_soma, 2,',','.').'</br>';

?>
```

Calculando os Totais do Formulário

- Utilizamos vários operadores nesta trecho de código.
- Também utilizamos a função *number_format()* para formatar os totais em Reais. Esta é uma função da biblioteca Math do PHP

Para maiores detalhes sobre a função *number_format()* acesse:
http://php.net/manual/pt_BR/function.number-format.php

ESTRUTURAS DE CONTROLE

Estruturas de Controle

- As estruturas de controle são as estruturas dentro de uma linguagem que permitem controlar o fluxo de execução por meio de um programa ou script. Podemos agrupá-las em estruturas condicionais (ou de desvio) e estruturas de repetição ou loops.

Tomando Decisões Condicionais

- Se quisermos responder sensatamente à entrada do nosso usuário, nosso código precisa ser capaz de tomar decisões. As construções que instruem nosso programa a tomar decisões são **chamadas condicionais**.

Instruções if

- Podemos utilizar uma instrução **if** para tomar uma decisão. Podemos fornecer à instrução if uma condição para utilização.
- Se a condição for *true*, o bloco de código seguinte será executado. As condições nas instruções if devem estar entre parênteses ( ).

Instruções if

- Se recebermos um pedido sem nenhum produto selecionado podemos fornecer uma mensagem amigável para o usuário.

```
if($total_qtd == 0) {

    echo '<p class="alerta">Informe a quantidade de produtos.</p>';
}
```

Para maiores detalhes sobre a instrução if acesse:
http://www.php.net/manual/pt_BR/control-structures.if.php

Instruções else

- Com frequência precisaremos decidir não apenas se desejamos realizar apenas uma ação, mas também um conjunto de possíveis ações desejamos realizar.
- Uma instrução **else** permite que uma ação alternativa seja tomada quando a condição em uma instrução **if** for *falsa*.

Instruções else

```php
<?php

    $aditivo_qtd = $_POST['aditivo_qtd'];
    $macaco_qtd  = $_POST['macaco_qtd'];
    $dvd_qtd     = $_POST['dvd_qtd'];

    $total_qtd  = 0;

    $total_qtd = $aditivo_qtd + $macaco_qtd + $dvd_qtd;

    if($total_qtd == 0) {

        echo '<p class="alerta">Informe a quantidade de produtos. </p>';

    } else {

    echo '<p>Pedido realizado com sucesso no dia ';
    echo date('d/m/y - h:i:s');
    echo '</p>';
    echo '<p>Resumo do pedido:</p>';

    echo 'Total de itens do pedido:  '.$total_qtd.'</br>';

    echo $aditivo_qtd . ' aditivo(s). </br>';
    echo $macaco_qtd . ' macacos hidraulico(s). </br>';
    echo $dvd_qtd . ' dvd(s).</br>';
```

Recuperação das variáveis do **formulário**

Início da instrução **if**

Final da instrução **if** e início da instrução **else**

Instruções else

```
    $total_soma = 0.00;

    define ('aditivo_preco', 17.10);
    define ('macaco_preco', 60.50);
    define ('dvd_preco', 629);

    $total_soma =  $aditivo_qtd * aditivo_preco
                 + $macaco_qtd  * macaco_preco
                 + $dvd_qtd     * dvd_preco;

    echo 'Subtotal do pedido: R$'.number_format($total_soma, 2,',','.').'</br>';

    $imposto = 0.10;
    $total_soma = $total_soma * (1 + $imposto);
    echo 'Total com impostos: R$'.number_format($total_soma, 2,',','.').'</br>';
}
?>
```

Final da instrução **else**

Instruções else

- Podemos construir processos lógicos mais complexos aninhando as instruções **if** uma dentro da outra.

```
if($aditivo_qtd > 0) {

    echo $aditivo_qtd . ' aditivo(s). </br>';
}
if($macaco_qtd > 0) {

    echo $macaco_qtd . ' macacos hidraulico(s). </br>';
}
if($dvd_qtd > 0) {

    echo $dvd_qtd . ' dvd(s).</br>';
}
```

Instruções elseif

- Para muitas das decisões que tomamos, há mais de duas opções. Podemos criar uma sequência de opções utilizando a instrução **elseif**.
- Esta instrução é uma combinação de uma instrução **else** e uma **if**.

Instruções elseif

- Suponhamos que temos a aplicação das seguintes regras de desconto para o pedido.
  - Menos que 10 aditivos comprados – nenhum desconto.
  - Entre 10 e 49 aditivos comprados – 5% de desconto.
  - Entre 50 e 99 aditivos comprados – 10% de desconto.
  - 100 ou mais aditivos comprados – 15% de desconto.

Instruções elseif

```
if($aditivo_qtd < 10) {
    $desconto_aditivo = 0;
}
elseif($aditivo_qtd >= 10 && $aditivo_qtd <= 49) {
    $desconto_aditivo = 0.5;
}
elseif($aditivo_qtd >= 50 && $aditivo_qtd <= 99) {
    $desconto_aditivo = 0.10;
}
elseif($aditivo_qtd >= 100) {
    $desconto_aditivo = 0.15;
}
```

Atividade

- Com base no trecho de código apresentado no slide anterior, calcule o total de desconto a ser aplicado no produto aditivo.

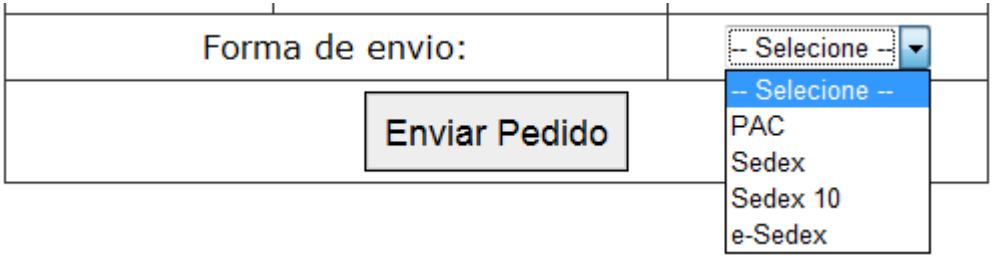
Instruções switch

- A instrução **switch** funciona de modo semelhante ao da instrução **if**, mas permite que a condição aceite mais de dois valores.
- Em uma instrução **if**, a condição pode ser *true* ou *false*. Em uma instrução **switch** a condição pode aceitar qualquer número de valores diferentes.
- É necessário fornecermos uma instrução **case** a fim de tratar cada valor para o qual deseja tomar uma ação.

ESTRUTURAS DE REPETIÇÃO

Instruções switch

- Iremos adicionar ao nosso formulário as formas de envio do produto para seleção do cliente.



Instruções switch

Item		Quantidade
	Aditivo para radiador R$17,10	☐
	Macaco hidráulico R $60,50	☐
	DVD automotivo R $629,00	☐
Forma de envio:		-- Selecione --
Enviar Pedido		

Instruções switch

- Inclua o código a seguir antes do final da instrução **else**.

Instruções switch

```php
$forma_envio = $_POST['forma_envio'];

switch($forma_envio) {

    case 'pac' :
        echo '<p> A forma de envio será via PAC.</p>';
    break;

    case 'sedex' :
        echo '<p> A forma de envio será via Sedex.</p>';
    break;

    case 'sedex10' :
        echo '<p> A forma de envio será via Sedex 10.</p>';
    break;

    case 'esedex' :
        echo '<p> A forma de envio será via e-Sedex.</p>';
    break;

    default :
        echo '<p> A forma de envio deve ser selecionada.</p>';
    break;

} // Final instrução switch
```

Instruções switch

- Quando um caso em uma instrução **switch** for ativado, o *PHP* executará instruções até alcançar uma instrução **break**.

Comparando as diferentes instruções condicionais

Qual instrução condicional utilizar?

- Não há nada que possamos fazer com uma ou mais instruções **else**, **elseif** e **switch** que não possamos fazer com um conjunto de instruções **if**.

Devemos tentar utilizar a instrução condicional que for mais legível na situação.

Atividade

- Adicione a instrução condicional switch para tratativa da forma de envio do pedido.

Iteração: Repetindo Ações

- Uma coisa em que os computadores sempre foram muito bons é automatizar tarefas repetitivas. Se houver algo que você precisa preparar da mesma maneira muitas vezes, pode utilizar um **loop** para repetir algumas partes do programa.

Loops while

- O tipo mais simples de loop no PHP é o loop **while**.
- Como uma instrução **if**, ele conta com uma condição. A diferença entre um loop **while** e uma instrução **if** é que uma instrução **if** executa uma vez o bloco de código se a condição for *true*. Um loop **while** executa o bloco repetidamente contanto que a condição seja **true**.

Loops while

- O seguinte *loop* while exibirá os números de 1 a 5.

```php
<?php

    $numero = 1;

    while($numero <= 5) {

        echo $numero."</br>";
        $numero++;
    }

?>
```

Loops while

- Exibiremos o custo do frete que será adicionado ao pedido do cliente dependendo da distância do destinatário.

Distância	Custo
50 km	R$ 5,00
100 km	R$ 10,00
150 km	R$ 15,00
200 km	R$ 20,00
250 km	R$ 25,00

Loops while

```
    } // Final instrução switch

    echo '<table width="400" border="1" cellspacing="2" cellpadding="5">';
    echo '<tr>';
    echo '<th>Distância</th>';
    echo '<th>Custo</th>';
    echo '</tr>';

    $distancia = 50;

    while($distancia <= 250) {

        $custo = $distancia / 10;

        echo '<tr>';
        echo '<td>' . $distancia . ' km</td>';
        echo '<td> R$ ' .  number_format($custo, 2,',','.'). '</td>';
        echo '</tr>';
        $distancia += 50;
    } // Final da instrução while

    echo '</table>';

} // Final instrução else
```

Loops for

- A maneira como utilizamos os loops **while** anteriormente é muito comum. Configuramos um contador com o qual iniciar. Antes de cada iteração, testamos o contador em uma condição. No final de cada iteração, modificamos o contador.
- Podemos escrever esse estilo de *loop* de uma forma mais compacta utilizando um *loop* **for**.

Loops for e foreach

```
    echo '<table width="400" border="1" cellspacing="2" cellpadding="5">';
    echo '<tr>';
    echo '<th>Distância</th>';
    echo '<th>Custo</th>';
    echo '</tr>';

    for($distancia = 50; $distancia <= 250; $distancia +=50) {

        $custo = $distancia / 10;

        echo '<tr>';
        echo '<td>' . $distancia . ' km</td>';
        echo '<td> R$ ' .  number_format($custo, 2,',','.'). '</td>';
        echo '</tr>';
    } // Final da instrução for

    echo '</table>';

} // Final instrução else
```

Podemos reescrever o exemplo de loop **while** como um loop **for**.
O loop for é mais compacto economizando duas linhas de código.

Atividade

- Substitua o loop while do exemplo anterior pelo loop for.

Interrompendo uma estrutura de controle

- Existem três abordagens para parar de executar uma parte de código.
- Para parar um **loop**, devemos utilizar a instrução ***break***. A execução do script continuará na próxima linha após o *loop*.
- Para pular para a próxima iteração do loop, utilizamos a instrução ***continue***.
- Para terminar de executar o script inteiro, utilizamos a instrução ***exit***.