

## 6. Estrutura de Controle

- Blocos
- Comandos condicionais
- Comandos de repetição

### 6.1. Blocos

São sequência de comandos delimitados por abre e fecha chaves ({ e })

### 6.2. Comandos Condicionais

**if**

- if (expressão)  
    comando;
  - if (expressão)  
    {  
        comando1;  
        ...  
        comandoN;  
    }
  - if (expressão)  
    comando1;  
    else  
        comando2;
  - if (expressão):  
    comando1;  
    ...  
    comandoN;  
    else  
        comando1;  
    ...  
    comandoN;
- endif**
- if(expressão1)  
    {  
        comando1;  
    }  
    elseif (expressão2)  
    {  
        comando2;  
    }

```
...  
else  
{  
    comandoN;  
}
```

O quadro 22 ilustra um exemplo do comando if.

## Quadro 22 – Exemplo do comando condicional if

```
<?php
$salario = 1000;
$desconto = 0.10; // 10%
$total = $salario - $salario*$desconto;
if($total>900) {
    echo"";
}
?>
```

**Switch**

```
switch (expressão)
{
    case valor1:
        comando1;
        break;
    case valor2:
        comando2;
        break;
}
```

O quadro 23 mostra um exemplo do comando switch.

## Quadro 23 – Exemplo do comando condicional switch

```
<?php
$i= 1;
switch($i) {
    case 0:
        print "i é iguala 0";
        break;
    case 1:
        print "i é iguala 1";
        break;
    case 2:
        print "i é iguala 2";
        break;
    default:
        print "i não é iguala 0, 1 ou 2";
        break;
}
?>
```

### 6.3. Comandos de repetição (laços)

- while
- do .. while
- for
- foreach

#### while

##### Sintaxe:

- while (expressão)  
{  
    comando;  
}
- while (expressão) :  
    comando1;  
    ...  
    comandoN;  
endwhile;

O quadro 24 mostra um exemplo de uso do comando while. O comando somente será executado se a condição for verdadeira.

Quadro 24 – exemplo de código com while

```
<?php  
$a = 1;  
while($a < 5) {  
    print$a;  
    $a++;  
}  
?>
```

**do..while****Sintaxe:**

```
do
{
    Comandos;
    ...
} while (condição);
```

O quadro 25 mostra um exemplo de uso do do..while. Caso a condição seja falsa, a repetição é encerrada.

Quadro 25 – Exemplo do do..while

```
$a=0;
do
{
    echo "a=$a\n";
    $a++;
}
while ($a<=10);
```

**for****Sintaxe :**

```
for (expr1; expr2; expr3) {
    comando1
    ..
    comandoN
}
```

**Parâmetros:**

- contador de inicialização: inicializa o valor do contador de loop
- contador de teste: avaliado para cada iteração do loop. Se for TRUE, o loop continua. Se for FALSO, o loop termina.
- contador de incremento: aumenta o valor do contador de loop

O quadro 26 mostra um exemplo de uso do comando for.

## Quadro 26 – Exemplo de uso do comando for

```
<?php
for ($x = 0; $x <= 10; $x++) {
    echo "O número é: $x <br>";
}
?>
```

**foreach**

O loop foreach - percorre um bloco de código para cada elemento em uma matriz.

Apresenta duas sintaxes:

- *foreach*(\$nome\_array **as** \$elemento)  
    {  
        comandos  
    }
- ou
- *foreach*(\$nome\_array **as** \$chave => \$valor)  
    {  
        comandos  
    }

Os quadros 13 e 16 já mostrados anteriormente ilustram exemplos de códigos com foreach.

**Comandos break e continue**

- **break**: pode ser usado em laços *do*, *for* e *while*. O break "quebra" a execução e continua executando o próximo comando. **break [n]**, onde n indica o número de estruturas que deverão ser finalizadas
- **continue**: interrompe a execução e avalia novamente a condição de teste. **continue [n]**, onde n indica o número de níveis que deverão ser iniciados.

O quadro 27 mostra dois exemplos, um exemplo com o break e outro com o continue.

Quadro 27 – Exemplo com os comandos break e continue

<pre>&lt;?php // break.php for(\$i=0; \$i &lt; 100; \$i++) {     if(\$i == 10) {         break;     }     echo" \$i "; } ?&gt;</pre>	<pre>&lt;?php // exibe os números pares for(\$i=0; \$i &lt; 100; \$i++) {     if(\$i % 2) {         continue;     }     echo" \$i " . '&lt;br&gt;'; } ?&gt;</pre>
--	---

## 7. Formulários HTML

O uso de formulário HTML (ou HTML form) é muito frequente na web, é um dos principais pontos de interação entre usuários e uma aplicação web. Geralmente, eles funcionam enviando os dados para um servidor, porém também podem ser utilizados para alguma função na própria página.

O formulário HTML é um formulário de preenchimento de dados ou que resulta em uma ação desejada utilizando a linguagem de marcação HTML. É formado por um ou mais widgets que podem ser por exemplo, campos de textos, caixas de seleção, botões, radio buttons e checkboxes. Dessa forma, o usuário pode interagir com a página ao executar ações através desses formulários.

O formulário HTML é representado pela tag de abertura **<form>** e a de fechamento **</form>**. Dentro dessas tags, serão colocados todos os elementos(widgets) que compõem este formulário.

Para a tag **<form>** podem ser atribuídos o atributo **method** e o atributo **action**. O atributo **action** define o local (através de uma URL) ao qual serão enviados todos os dados recolhidos do formulário. Já o atributo **method** define o método HTTP com que o formulário HTML irá lidar com os dados recebidos. São eles: o método **GET** e o método **POST**.

O protocolo HTTP (HyperText Transfer Protocol) utiliza vários métodos de manipulação e organização dos dados. Os dois métodos mais utilizados para submeter dados de formulários são o **GET** e o **POST**.

Ambos os métodos transferem dados do browser para o servidor, a maior diferença entre eles é a maneira como essa informação é transmitida

#### GET

- O browser acrescenta ao URL, especificado no atributo ACTION, um "?" e os valores codificados;
- Os dados não são encriptados, logo informações que exigem segurança não devem ser manipuladas por este método;
- Suporta apenas até 128 caracteres, logo é útil para valores pequenos.

#### POST

- Os dados introduzidos num formulário fazem parte do corpo da mensagem enviada para o servidor;
- Pode encriptar os dados;
- É possível transferir uma grande quantidade de dados.
- Este é o método aconselhado.

Os quadros 28 e 29 mostram um exemplo de uso do método POST e o quadro 30 do método GET.

Quadro 28 – Exemplo de uso do método POST usando o atributo action

form.html	script.php
<pre>&lt;form method="POST" action="script.php"&gt;   Nome: &lt;input name="nome" type="text"&gt;   &lt;input type="submit"&gt; &lt;/form&gt;</pre>	<pre>&lt;?php   \$nome=\$_POST["nome"];   echo "&lt;p&gt;nome=\$nome"; ?&gt;</pre>

Quadro 29 – Exemplo de uso do método POST sem o uso do atributo action

form.html	script.php
<pre>&lt;form method="POST"&gt;   Nome: &lt;input name="nome" type="text"&gt;   &lt;input type="button" value="ok"   onclick="script.php"&gt; &lt;/form&gt;</pre>	<pre>&lt;?php   \$nome=\$_POST["nome"];   echo "&lt;p&gt;nome=\$nome"; ?&gt;</pre>

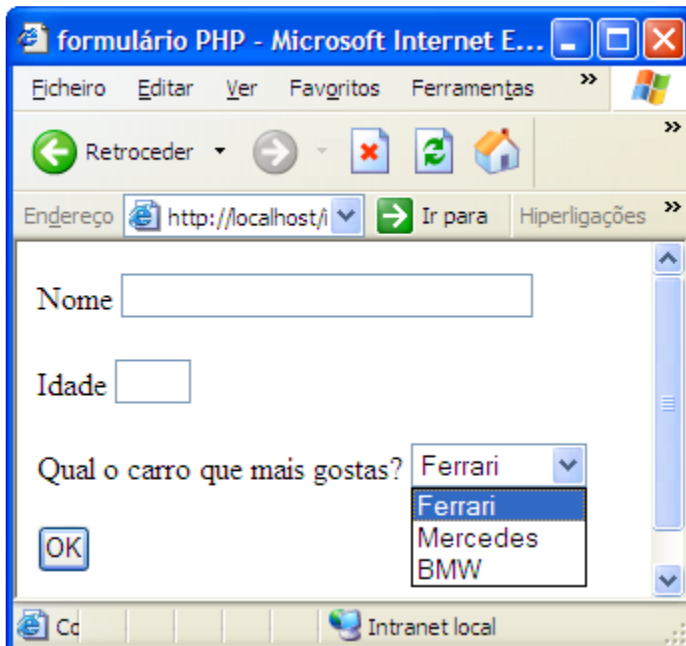


Quadro 30 - Exemplo de uso do método GET

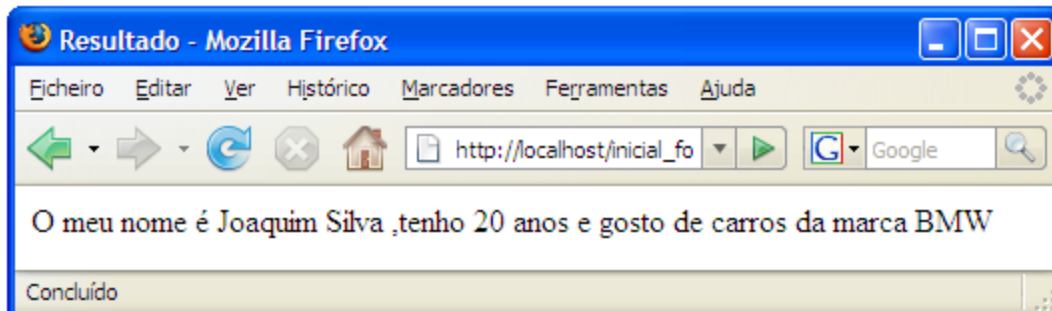
form.html	script.php
<pre>&lt;formmethod="GET" action="script.php"&gt;   Nome: &lt;input name="nome" type="text"&gt;   &lt;input type="submit"&gt; &lt;/form&gt;</pre>	<pre>&lt;?php   \$nome=\$_GET["nome"];   echo "&lt;p&gt;nome=\$nome"; &lt;/php&gt;</pre>

### Exercício

Dado o formulário:



Faça o código necessário para enviar os dados para o servidor que por sua vez irá responder com o formulário abaixo. Utilize o método POST.



Neste exemplo, o usuário digitou Joaquim Silva no nome, 20 na idade e escolheu a marca BMW.