

MÓDULO

PROGRAMAÇÃO PHP

UNIDADE

ESTRUTURAS DE CONTROLO E EXCEÇÃO
EM PHP

ÍNDICE

OBJETIVOS.....	3
INTRODUÇÃO.....	4
1. ESTRUTURAS DE CONTROLO	5
1.1. CONDIÇÃO IF.....	5
1.2. WHILE E DO-WHILE.....	7
1.2.1. WHILE.....	7
1.2.2. DO-WHILE.....	7
1.3. FOR E FOREACH.....	8
1.3.1. FOR.....	8
1.3.2. FOREACH.....	8
1.4. BREAK E CONTINUE	9
1.4.1. BREAK.....	9
1.4.2. CONTINUE	10
1.5. SWITCH.....	10
1.6. INCLUDE/INCLUDE_ONCE/REQUIRE/REQUIRE_ONCE	11
2. GESTOR DE EXCEÇÕES.....	14
CONCLUSÃO.....	17
AUTOAVALIAÇÃO	19
SOLUÇÕES.....	23
PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO.....	24
BIBLIOGRAFIA	25

OBJETIVOS

Com esta unidade didática, pretende-se que desenvolva os seguintes objetivos de aprendizagem:

- Utilizar estruturas de controlo.
- Utilizar o manipulador de erros.

INTRODUÇÃO

A ferramenta mais utilizada nas linguagens de programação são as estruturas de controlo, pois estas instruções da linguagem permitem fazer perguntas ao código, ou seja, comparar operadores com um valor e executar uma opção dependendo do valor.

1. ESTRUTURAS DE CONTROLO

As estruturas de controlo permitem controlar o rumo do programa, tomar decisões, realizar ações repetitivas, etc., dependendo das condições estabelecidas.

As principais estruturas de controlo do PHP permitem condicionar a execução de um código ao resultado de uma comparação entre variáveis ou valores.

1.1. CONDIÇÃO IF

A condição if é a mais utilizada. Esta condição cria um valor boolean de uma dada expressão e, se o valor devolvido for true, ela executa o código encontrado na mesma. Segue-se um exemplo da sua estrutura.

```
if ($var > $var2){  
    echo "declaração a executar.";  
}
```

Neste exemplo, a condição verifica se o valor da variável \$var é maior que o valor da variável \$var2 e, se for verdadeiro, executa o código dentro das chavetas.

Esta declaração pode ser seguida por outra chamada else que surge sempre após o if. Permite executar um código diferente se o valor devolvido for false. Segue-se um exemplo para clarificar:

```
$var = 2;
$var2 = 4;
if ($var > $var2){
    echo 'var é maior que var2';
}else{
    echo 'var é menor ou igual a var2';
}
```

Por fim, há um método para concatenar várias instruções if chamado elseif que permite executar um pedaço de código após um if, se o valor devolvido pela condição inicial for falso (desde que a condição elseif seja verdadeira). Segue-se um exemplo:

```
$var = 2;
$var2 = 4;
if ($var > $var2){
    echo 'var é maior que var2';
}elseif ($var = $var2){
    echo 'var é igual a var2';
}else{
    echo 'var é menor que var2';
}
```

Neste exemplo, verifica-se se a condição do if é falsa e verifica-se a condição do elseif, que, se também for falsa, levará à execução do else. No entanto, poderá acrescentar o número de elseif que pretender.

1.2. WHILE E DO-WHILE

1.2.1. WHILE

Os loops são instruções mais complexas. Começando pela mais simples, a declaração while indica ao PHP para executar um código que está entre chavetas até que a condição while devolva false. A sua sintaxe:

```
$var = 0;
while ($var < 10){
    echo $var;
    $var ++;
}
```

O exemplo anterior escreve no ecrã e aumenta a variável \$var em um valor, desde que o seu valor seja menor que dez.

1.2.2. Do-WHILE

Este loop é muito parecido com o anterior, a única coisa diferente é que este verifica o valor devolvido no final do mesmo, ou seja, o código contido no loop é executado pelo menos uma vez antes que o comando seja validado e repetido. Um exemplo:

```
$i = 0;
do {
    echo $i;
} while ($i > 0);
```

O loop anterior seria executado apenas uma vez, embora a declaração devolva false desde o início.

1.3. FOR E FOREACH

1.3.1. FOR

Estes loops são mais complexos do que os while, pois recebem três parâmetros durante a sua criação:

- O primeiro parâmetro é executado uma vez no início do loop e serve para inicializar uma variável que servirá como contador de um determinado valor.
- O segundo parâmetro é aquele que é avaliado em cada rodada do loop desde o início e, se devolver true, o código será executado dentro do loop, caso contrário, o loop será encerrado.
- O terceiro é avaliado cada vez que um loop termina e é frequentemente utilizado para alterar o valor da primeira variável de parâmetro.

Segue-se um exemplo para esclarecer o uso deste loop:

```
for ($i = 1; $i <= 10; $i++) {  
    echo $i;  
}
```

Com o código anterior, são escritos os números de 1 a 10 na página web.

1.3.2. FOREACH

Este loop é utilizado para interagir com arrays e iteráveis e só funciona com esses tipos de dados. Basicamente, permite percorrer um array sem a necessidade de calcular os elementos ou carregar contadores. Existem duas maneiras de serem utilizados:

- A primeira forma passa pelo array, atribuindo, a cada ciclo do loop, o valor do elemento do array à variável indicada e avançando automaticamente na posição. Por exemplo:

```
$vector = array("laranja", "azul", "verde");  
foreach ($vector as $var) {  
    echo "Valor: $var<br />\n";  
}
```

- A segunda forma permite, além de conhecer o valor de cada elemento do vetor, conhecer a sua chave:

```
$vector = array("laranja", "azul", "verde");  
foreach ($vector as $chave => $var) {  
    echo "Chave: $chave Valor: $var) />\n";  
}
```

1.4. BREAK E CONTINUE

1.4.1. BREAK

Esta instrução é utilizada para encerrar a execução da mesma em qualquer ponto do código dentro de uma instrução for, foreach, while ou switch; para fazer isso, basta chamá-lo e a execução irá parar e saltar fora do loop. Por exemplo:

```
for ($i = 1;$i < 5;$i++){  
    echo $i;  
    if ($i == 3){  
        break;  
    }  
}
```

O exemplo anterior executa um loop for escrevendo os números de um a três no ecrã e, quando chega a três, sai da execução do for.

1.4.2. CONTINUE

Esta instrução é utilizada para saltar o resto do código que falta executar e ir diretamente para a verificação da instrução, sem sair do loop. Por exemplo:

```
for ($i =1;$i<=5;$i++){  
    if ($i == 3){  
        continue;  
    }  
    echo $i;  
}
```

Este exemplo mostra no ecrã os números de um a cinco, exceto o número três, uma vez que, quando a variável da condição for três, o continue é executado, evitando a instrução echo.

1.5. SWITCH

Esta instrução é semelhante a juntar muitas condições if; ajuda a comparar uma variável a muitos valores e, dependendo do valor, executar um ou outro código. A comparação pode ser feita com números ou strings. Cada comparação é seguida por um case e poderá incluir uma comparação padrão, chamada default, que será executada se não devolver true em qualquer um dos case. Segue-se um exemplo:

```
$i = 7;  
switch ($i) {  
    case 0:  
        echo "i é igual a 0";  
        break;  
    case 1:  
        echo "i é igual a 1";  
        break;  
    case 2:
```

```
echo "i é igual a 2";  
break;  
default:  
echo "i não é igual a 0, 1 nem 2";  
}
```

No exemplo anterior, o texto “i não é igual a 0, 1 ou 2” será exibido na tela, uma vez que as restantes comparações não são cumpridas.

1.6. INCLUDE/INCLUDE_ONCE/REQUIRE/REQUIRE_ONCE

O `include` permite inserir um ficheiro completo no código da página web. Este ficheiro pode conter código de programação web de qualquer tipo: HTML, PHP, JavaScript, entre outros. Os ficheiros são incluídos a partir do caminho de chamada (se estiverem num caminho diferente, deve ser criado com “..” ou “/”) e geralmente são utilizados para carregar excertos de código que são repetidos em todas as páginas web, como o cabeçalho da página, o menu ou um ficheiro PHP com as funções comuns.

Quando um ficheiro é incluído, o código dentro dele é afetado em relação às variáveis, como se fossem colocadas diretamente no ficheiro pai.

É extremamente útil para poupar trabalho e espaço, enquanto o código da página fica mais claro, pois está distribuído.

- **include**: pode ser utilizado várias vezes para incluir partes de código, por exemplo categorias de uma loja.
- **include_once**: usado para incluir uma parte do código apenas uma vez.
- **require**: exige a inclusão do ficheiro, caso contrário, o código irá parar de criar um erro fatal de PHP.
- **require_once**: só pode ser adicionado uma vez.

Exemplo

Neste exemplo irá carregar-se, com a instrução `include`, um ficheiro que contém o carregamento de várias variáveis. Para este exemplo, serão utilizados dois ficheiros:

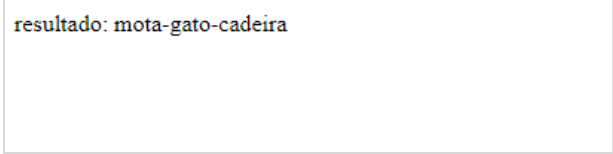
- `variaveis.php` (ficheiro externo):

```
<?php
$variavel = 'mota';
$variavel2 = 'gato';
$variavel3 = 'cadeira';
?>
```

- `exemplo.php`:

```
<html>
<body>
<p>
<?php
include 'variaveis.php';
echo 'resultado: '.$variavel.'-'.$variavel2.'-'.$variavel3;
?>
</p>
</body>
</html>
```

A visualização da página será:



```
resultado: mota-gato-cadeira
```

A outra instrução que ajuda a carregar ficheiros é o `require`, que faz exatamente o mesmo que o `include`, porém, enquanto como `include`, se um erro for criado, um aviso é produzido e o script continuará a sua execução, ao utilizar `require` o script irá parar.

Por fim, existe o `require_once`. A diferença em relação ao `require` é que, se o ficheiro já foi incluído anteriormente na página, não será incluído novamente.

2. GESTOR DE EXCEÇÕES

Tal como em muitas outras linguagens de programação, o PHP possui um manipulador de exceções. Uma exceção pode ser lançada e capturada dentro da execução do PHP: o código terá de estar dentro de um try e cada try deverá ter pelo menos um catch, e se nenhum erro for criado ou lançar qualquer exceção na execução do script, continuará após o último catch. É uma forma de controlar a execução de um código específico inserindo manipuladores de erro definidos.

Segue-se um exemplo para esclarecer a sua utilização:

Exemplo

Neste exemplo, pretende-se verificar vários dados sobre um valor passado como parâmetro para uma função, mais especificamente se é igual a vazio (''), se é igual a "carros" e se é igual a "motas".

```
<html>
<body>
<p>
<?php
function teste($var){
try {
if ($var=='') {throw new Exception('não estou a pensar em nada');}
```



```
if ($var=='carros') {throw new Exception('estou a pensar em
carros');}

if ($var=='motas') {throw new Exception('estou a pensar em
motas');}

echo 'pensei em '.$var;
} catch (Exception $e) {
    echo $e->getMessage();
}
}

teste('motas');

?>

</p>
</body>
</html>
```

A visualização será:

estou a pensar em motas

CONCLUSÃO

As estruturas de controlo são o ponto forte da linguagem PHP, uma vez que, graças a elas, é possível controlar a execução do código, a entrada de dados, a criação de conteúdo, basicamente tudo o que pretender para a página web.

AUTOAVALIAÇÃO

1. **O que é que a condição if faz?**
 - a) Executa um código, se a condição for verdadeira.
 - b) Executa um código até que uma condição seja verdadeira.
 - c) Executa uma condição tantas vezes quantas forem indicadas no contador anterior.
 - d) Executa uma condição dependendo do código.
2. **Quando é que o código dentro de um else é executado?**
 - a) Quando a expressão dentro de if devolve true.
 - b) Quando a expressão dentro de if devolve false.
 - c) Nunca funciona.
 - d) Quando a sua própria expressão devolve true.
3. **O que é utilizado para verificar uma expressão após um valor false de um if sem sair da instrução?**
 - a) repeat.
 - b) elseif.
 - c) ifelse.
 - d) forif.

4. Qual é a diferença entre o while e o do while?

- a) O while é utilizado em funções.
- b) O while executa o código contido pelo menos uma vez.
- c) O while nunca executa o código contido mais de cinco vezes.
- d) O do while executa o código contido pelo menos uma vez.

5. Complete a afirmação: o primeiro parâmetro de um for...

- a) ... é utilizado para declarar o valor inicial de uma variável.
- b) ... é utilizado para avaliar cada volta do loop.
- c) ... é avaliado a cada volta e o valor da variável avaliada geralmente muda para continuar dentro do loop ou sair.
- d) ... é utilizado para identificar o for.

6. Qual é a instrução que permite percorrer um array e é usada exclusivamente para isso?

- a) for.
- b) foritem.
- c) foreach.
- d) switch.

7. O que é utilizado para comparar um valor com uma série de resultados e agir de forma diferente para cada resultado utilizando uma única instrução?

- a) switch.
- b) elseif.
- c) for.
- d) foreach.

8. Qual é a diferença entre o require e o require_once?

- a) O require_once verifica se o ficheiro já foi carregado e, em caso afirmativo, recarrega-o.
- b) O require_once carrega onze linhas de código de um ficheiro.
- c) O require_once é utilizado apenas para carregar ficheiros .php.
- d) O require_once carrega um ficheiro externo apenas na primeira vez.

9. Quais são os termos utilizados pelo manipulador de exceções em PHP?

- a) try error.
- b) try exception.
- c) try catch.
- d) catch resume.

10. O que é que a expressão continue dentro de um for indica?

- a) A saída da execução do for.
- b) A execução do continue sem variações.
- c) Que se salte o resto do código dentro do for e se verifique novamente a expressão do loop.
- d) Que se torne o for num loop infinito.

SOLUÇÕES

1.	a	2.	b	3.	b	4.	d	5.	a
6.	c	7.	a	8.	d	9.	c	10.	c

PROPOSTAS DE DESENVOLVIMENTO DO ESTUDO

Para mais informação visite os seguintes websites:

- https://www.w3schools.com/php/php_if_else.asp (em inglês).
- https://www.w3schools.com/php/php_switch.asp (em inglês).
- https://www.php.net/manual/pt_BR/language.control-structures.php
- https://www.php.net/manual/pt_BR/language.exceptions.php

BIBLIOGRAFIA

- Cabezas, L. M. (2010), *Php 5*. Madrid: AnayaMultimedia.

