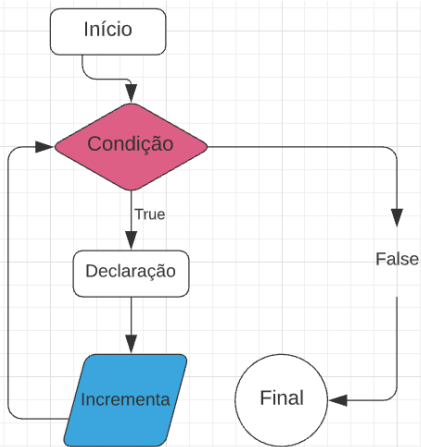


Laços de repetição em PHP



Prof. Wanderlei Silva do Carmo
wander.silva@gmail.com



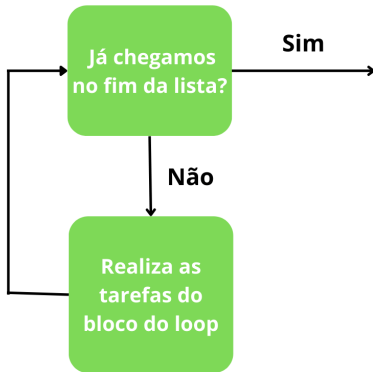


Introdução

Laços de repetição em PHP permitem que um bloco de código seja executado várias vezes. Esses laços são essenciais para a maioria dos projetos em PHP, pois permitem que o código seja executado de forma mais eficiente e econômica em termos de tempo e recursos. Existem três tipos principais de laços em PHP: **for**, **while** e **do-while**.

Laço for

O laço **for** é usado quando se sabe exatamente quantas vezes o bloco de código precisa ser executado. Ele consiste em uma inicialização, uma condição e uma atualização. O laço for é uma das estruturas de repetição mais usadas em PHP.



For Loop

O `for` loop é outra estrutura de repetição que permite executar um bloco de código repetidamente. O `for` loop é frequentemente usado quando se sabe exatamente quantas vezes o loop deve ser executado.

O formato básico do `for` loop é:

php

 Copy code


```
for (inicialização; condição; incremento) {  
    // código a ser executado  
}
```

Onde:

- `inicialização`: é a expressão que é executada antes do primeiro loop. Normalmente é usada para inicializar uma variável de contagem.
- `condição`: é a expressão que é verificada no início de cada iteração do loop. O loop continua enquanto a condição for verdadeira.
- `incremento`: é a expressão que é executada no final de cada iteração do loop. Normalmente é usada para incrementar a variável de contagem.

Exemplo:

php

 Copy code

```
for ($i = 1; $i <= 5; $i++) {  
    echo $i . " ";  
}  
// Resultado: 1 2 3 4 5
```

Este exemplo usa um `for` loop para exibir os números de 1 a 5. A variável `$i` é inicializada em 1, o loop continua enquanto `$i` for menor ou igual a 5, e `$i`

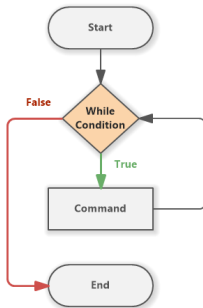
é incrementada em 1 a cada iteração do loop.

Laço while/do while

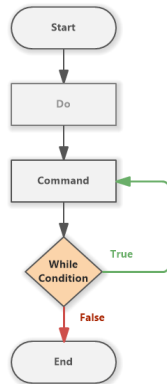
O laço **while** é usado quando não se sabe quantas vezes o bloco de código precisa ser executado. Ele é executado enquanto a condição especificada for verdadeira. O laço while é útil quando se precisa executar o código até que uma determinada condição seja atendida.

O laço **do-while** é semelhante ao laço while, mas a diferença é que o bloco de código é executado pelo menos uma vez, independentemente da condição. A condição é verificada após a primeira execução, e se a condição for verdadeira, o bloco de código é executado novamente.

WHILE




DO-WHILE



While Loop

O `while` loop é uma estrutura de repetição que executa um bloco de código enquanto uma determinada condição é verdadeira. O formato básico do `while` loop é:

php


 Copy code

```
while (condição) {  
    // código a ser executado  
}
```

O código dentro do `while` loop é executado repetidamente enquanto a condição especificada for verdadeira. A condição é verificada no início de cada iteração do loop.

Exemplo:

php

 Copy code

```
$i = 1;

while ($i <= 5) {
    echo $i . " ";
    $i++;
}

// Resultado: 1 2 3 4 5
```


Este exemplo usa um `while` loop para exibir os números de 1 a 5. A variável `$i` é incrementada a cada iteração do loop, e o loop continua enquanto `$i` for menor ou igual a 5.

Do-While Loop

O `do-while` loop é semelhante ao `while` loop, mas a condição é verificada no final de cada iteração, o que significa que o bloco de código será executado pelo menos uma vez, mesmo que a condição inicialmente seja falsa.

O formato básico do `do-while` loop é:


php

 Copy code

```
do {  
    // código a ser executado  
} while (condição);
```

Exemplo:

php

 Copy code

```
$i = 1;


do {
    echo $i . " ";
    $i++;
} while ($i <= 5);
// Resultado: 1 2 3 4 5
```

Este exemplo é semelhante ao exemplo anterior, mas usa um `do-while` loop em vez de um `while` loop. O loop é executado pelo menos uma vez, mesmo que a condição inicial seja falsa.

Foreach

A estrutura foreach é usada para percorrer arrays e objetos. Ela executa um conjunto de instruções para cada elemento do array ou propriedade do objeto.

php

 Copy code

```
$cores = array("vermelho", "verde", "azul");  
foreach ($cores as $cor) {  
    echo $cor;  
}
```

Neste exemplo, o loop percorre o array `$cores` e a cada iteração, a variável `$cor` recebe o valor do próximo elemento do array. O loop é executado uma vez para cada elemento do array.

Essas são algumas das estruturas de repetição em PHP que podem ser usadas para repetir um conjunto de instruções várias vezes. É importante entender como cada uma delas funciona e em que situações podem ser mais adequadas.

Break e Continue



Além das estruturas de repetição básicas, PHP oferece duas instruções que podem ser usadas dentro de loops para controlar o fluxo do programa: **break** e **continue**. A instrução **break** é usada para sair de um loop imediatamente. Se a instrução **break** for encontrada dentro de um loop, o loop será interrompido imediatamente e o programa continuará a partir da próxima linha de código após o loop.

break

A instrução **'break'** é usada para sair de um loop imediatamente. Se a instrução **'break'** for encontrada dentro de um loop, o loop será interrompido imediatamente e o programa continuará a partir da próxima linha de código após o loop.

Exemplo:

```
php Copy code

$i = 1;

while (true) {
    echo $i . " ";

    if ($i >= 5) {
        break;
    }

    $i++;
}

// Resultado: 1 2 3 4 5
```

Este exemplo usa um **'while'** loop para exibir os números de 1 a 5, mas a condição do loop é definida como **'true'**, o que significa que o loop continuará indefinidamente. A instrução **'break'** é usada para sair do loop e o resultado é igual a 5.

continue

A instrução `continue` é usada para pular uma iteração do loop e continuar com a próxima. Se a instrução `continue` for encontrada dentro de um loop, o código dentro do loop para de ser executado para a iteração atual e o loop continua com a próxima iteração.

Exemplo:

```
php Copy code

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

    echo $i . " ";
}

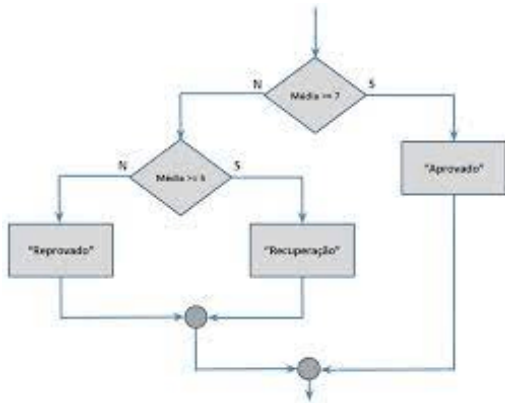
// Resultado: 1 2 4 5
```

Este exemplo usa um `for` loop para exibir os números de 1 a 5, mas a instrução `continue` é usada para pular a iteração quando `$i` é igual a 3. Isso significa que o número 3 não será exibido no resultado final.

Laços aninhados

Os laços **for**, **while** e **do-while** podem ser aninhados para criar estruturas mais complexas.

Laços de repetição aninhados são loops que estão aninhados um dentro do outro e ambos são executados em conjunto.



Neste exemplo, o loop externo percorre os números de 1 a 10, e o loop interno também percorre os números de 1 a 10. Dentro do loop interno, os números são multiplicados e o resultado é exibido. Em seguida, uma quebra de linha é adicionada para separar as linhas da tabela de multiplicação.

Os laços de repetição aninhados podem ser úteis em várias situações onde você precisa repetir um conjunto de instruções várias vezes em diferentes níveis de aninhamento. É importante ter cuidado ao usar laços de repetição aninhados, pois eles podem ser difíceis de entender e depurar, especialmente quando há muitos níveis de aninhamento. Por isso, é importante manter o código limpo e bem organizado, e usar nomes de variáveis descritivos para facilitar a leitura e o entendimento do código.

Aqui está um exemplo de como isso pode ser feito em PHP usando laços de repetição aninhados:

php

 Copy code

```
// loop externo que percorre os números de 1 a 10
for ($i = 1; $i <= 10; $i++) {
    // loop interno que percorre novamente os números
    for ($j = 1; $j <= 10; $j++) {
        // multiplica os números e exibe o resultado
        echo $i . " x " . $j . " = " . ($i * $j) . "<br>";
    }
    // adiciona uma quebra de linha após cada linha d
    echo "<br>";
}
```

Neste exemplo, o loop externo percorre os números de 1 a 10, e o loop interno também percorre os números de 1 a 10. Dentro do loop interno, os números são multiplicados e o resultado é exibido. Em seguida, uma quebra de linha é adicionada para separar as linhas da tabela de multiplicação.

Conclusão

Os laços de repetição são uma parte fundamental da programação em PHP. Eles permitem que o código seja executado várias vezes, economizando tempo e recursos. É importante escolher o tipo certo de laço para cada situação, e evitar laços aninhados complexos sempre que possível.