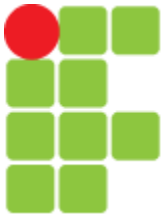

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Programação Web

Aula 05 – Estruturas de controle em PHP

Prof. Pedro Baesse
pedro.baesse@ifrn.edu.br

Roteiro

- Comandos Condicionais
- Comandos de Repetição
- Comando de Fluxo de Execução

Estruturas de controle em PHP

- Comandos usados para estruturar seus programas
- Comandos comuns a maioria das linguagens de programação
- Uso fundamental para realizar decisões lógicas, testar se determinada expressão é verdadeira, repetir um bloco de comandos por um certo número de vezes ou até que uma condição seja atingida

Comandos Condicionais

- Podemos avaliar uma expressão e, dependendo do resultado obtido, executar um trecho de código diferente
- Usado na tomada de decisão dentro de um programa
- Exemplo imprimir o valor aprovado caso a nota do aluno seja maior que 6, senão imprimir reprovado

Comandos Condicionais

- if
- switch

if

- Avalia uma expressão e dependendo do resultado é executado um conjunto diferente de instruções

```
if  ( exp1 )
        { bloco1 }

elseif  ( exp2 )
        { bloco2 }

else
        { bloco3 }
```

- Se a exp1 for verdadeira, execute o bloco1
- Senão se exp2 for verdadeira, execute o bloco2
- Senão execute o bloco3

if

- Lembrar que somente um dos blocos será executado. Depois disso a execução continuará depois do comando if
- If em português significa “se” e o else significa “senão”.
- Pode aparecer diversos elseif
- Caso o bloco só tenha uma linha chaves ({}) são dispensáveis
- Não é obrigatório o uso do elseif ou else. O if isoladamente também pode ser usado

if – Exercício

```php
$prova1 = 7;
$prova2 = 5;
$nota = ($prova1 + $prova2) / 2;

if($nota<3)
        $desempenho ="PÉSSIMO";
elseif($nota<5)
        $desempenho ="RUIM";
elseif($nota<7)
        $desempenho ="MÉDIO";
elseif($nota<8)
        $desempenho ="BOM";
else
        $desempenho ="EXCELENTE";

echo "O seu desempenho foi $desempenho";
```

if

- Outra sintaxe alternativa do if é o uso do endif, para determinar o fim de um comando

```
if  ( exp1 ):
        bloco1
elseif  ( exp2 ):
        bloco2
else:
        bloco3
endif;
```

- Não é necessário o uso das chaves pois o PHP interpreta desde os dois pontos (:) até o próximo elseif, else ou endif

Switch

- Parecido com o if, pois ambos avaliam o valor de uma expressão para escolher o que vai ser executado
- Quando se tem a mesma variável com valores diferentes valores para ser avaliado, já que usa basicamente a igualdade e o if qualquer condição

Switch

- Sintaxe mais clara e organizada que o if

```
switch ( operador )
{
        case valor1:
                <comandos>
                break;

        case valor2:
                <comandos>
                break;

        case valorN:
                <comandos>
                break;
        default:
                 <comandos>
                break;
}
```

- Depois de cada bloco de comandos, deve ser usado o comando break para o switch seja encerrado. Caso não seja usado o PHP continuará executando o switch

if vs switch

- Uso do if

```php
$numero = 2;

if($numero == 0){

        echo "O númeor é 0<br>";
}

elseif($numero == 1){

        echo "O númeor é 1<br>";
}

elseif($numero == 2){

        echo "O númeor é 2<br>";
}
```

- Uso do switch

```php
$numero = 2;

switch($numero){

        case 0:
                echo "O númeor é 0<br>";
                break;

        case 1:
                echo "O númeor é 1<br>";
                break;

        case 2:
        echo "O númeor é 2<br>";
        break;
}
```

Switch

```
$opcao = "";
switch($opcao){

        case 's':

                echo "Você escolheu SIM!";
                break;

        case 'n':
                echo "Você escolheu NÃO!";
                break;

        default:
                echo "A opção inválida";
                break;
}
```

- A opção default funciona como o else do comando if. Caso todas as outras alternativas sejam falsas, ele será executado
- Também é possível usar condições alfanuméricas

Comandos de repetição

- Utilizados para que um bloco de instruções seja executado por um número determinado de vezes, ou até que uma condição seja atingida

Comandos de repetição

- while
- do...while
- for
- for each

while

- Traduzido para o português significa enquanto
- Composto por uma expressão e um bloco de comando
- O comando avalia a expressão, e enquanto essa expressão retornar o valor verdadeiro, a execução do conjunto de comandos será repetida. Caso seja falsa o bloco encerra a execução do bloco
- Tomar cuidado para não criar expressões que nunca se tornam falsas pois teríamos um loop infinito.

while

- Sintaxe

```
while ( exp )
{
        <comandos>
}
```

- Sintaxe alternativa

```
while ( exp ):
        <comandos>
endwhile;
```

while – Exercício

```
while($cont<100)
{
        echo "O valor atual do contador é $cont <br>";
        $cont++;
}
```

- O que acontece se ao inves de $cont<100 tivessemos $cont!=0?

do...while

- A única diferença entre o while e o do...while é que o while avaliza a expressão no início do laço e o do...while ao final
- Vai ser executado ao menos uma vez e caso usasse o while não seria executado nenhuma vez
- Sintaxe

```
do
{
        <comandos>
} while ( exp )
```

do...while – Exercício

```php
$numero = 1;

do
{
        echo "O valor atual do contador é $cont <br>";
        $numero++;

}while($numero<15);
```

for

- Usado quando queremos executar um conjunto de instruções por quantidade especifica de vezes
- Pode ser usado para imprimir os elementos de um array ou todos os resultados de uma consulta no banco de dados

for

- Sintaxe

```
for (inicialização ; condição; operador)
{
        <comandos>

}
```

- Sintaxe Alternativa

```
for (inicialização ; condição; operador):

        <comandos>

endfor;
```

for

- Com *inicialização* iniciamos o valor inicial da variável que controlará o loop
  - *$cont = 0;*
- Na *condição* devemos colocar a condição para que o loop continue a ser executado. Quando a condição retornar um valor falso o loop parará
  - $cont<20
- O *operador* é usado para atualizar o valor da varíavel de controle, fazendo um incremento ou decremento ao final de cada iteração do loop
  - $cont++

for – Exercício

```
echo "Contagem Progressiva <br> <br>";

for($cont=0;$cont<10;$cont++){

        echo "A variável \$cont vale $cont <br>";

}

echo "<br> Contagem Regressiva <br> <br>";

for($cont=13;$cont>0;$cont--){

        echo "A variável \$cont vale $cont <br>";

}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

for

- Também é possível fazer loops aninhados
- Útil para um array bidimensional

```
$vetor[0][0] = "elemento00";
$vetor[0][1] = "elemento01";
$vetor[1][0] = "elemento10";
$vetor[1][1] = "elemento11";

for($contI=0;$contI<2;$contI++){

        for($contJ=0;$contJ<2;$contJ++){

                echo "O valor do vetor é " . $vetor[$contI][$contJ];
                echo "<br>";
        }
}
```

foreach

- Oferece uma maneira mais fácil de “navegar” entre os elementos de um array

```
foreach($nome_array as $elemento)
{
        <comandos>
}
```

- Todos os itens de *$nome_array* serão visitados. A cada iteração o item da vez será armazenado em *$elemento.* Assim é possível trabalhar todos os elementos usando somente uma variável

foreach

- Essa segunda sintaxe funciona da mesma forma porém enquanto o elemento é adicionado *$valor,* o índice atual é atribuído a *$chave*

```
foreach($nome_array as $chave => $valor)
{
        <comandos>
}
```

foreach

- Essa segunda sintaxe funciona da mesma forma porém enquanto o elemento é adicionado *$valor,* o índice atual é atribuído a *$chave*

```
foreach($nome_array as $chave => $valor)
{
        <comandos>
}
```

foreach – Exercício

```
$vetor = arrav (1.2.3.4,5);
foreach($vetor as $v)
{
        print "O valor atual do vetor é $v. <br>";
}

$a = arrav ("um"=>1. "dois"=>2. "tres"=>3);
foreach($a as $chave => $valor)
{
        print("\$a[$chave] => $valor.<br>");

}
```

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Comandos de Controle de Fluxo

- Existem comandos que podem ser usados juntamente com as estruturas vistas
  - break
  - continue

Break

- Termina a execução do comando atual, podem ser um if, for, while ou switch. O fluxo continua exatamente no primeiro comando após a estrutura

Break – Exercício

```php
$vetor = array (1,2,3,4,5,6,7,8,9,10);
$k = 0;

while($k < 10)
{
    if($vetor[$k]=="sair")
    {
        break;
    }
    echo $vetor[$k] . "<br>";

    $k++;
}
```

Break

- O comando break também aceita um argumento numérico opcional, que informa quantas estruturas devem ser finalizadas

Break – Exercício

```php
$k = 0;
$i = 0;
while($k < 10)
{
        $i++;
        $k++;
        while($i<20)
        {
                if($i==10)
                {
                        echo "Encerrando o primeiro while <br>";
                        break;
                        echo "Essa linha nunca vai ser impressa";
                }
                elseif($i==15)
                {
                        echo "Encerrando os 2 whiles...";
                        break 2;
                }
                $i++;
        }
}
```

Continue

- Usado para ignorar o restante das instruções dentro do comando de repetição indo para a próxima iteração (voltando para o início do laço)
- Também aceita o argumento numérico opcional, voltando a execução para número especificado de estruturas

Continue – Exercício

```
$vetor = arrav (1.3.5.7.8.11.12.15,20);
for($i=0; $i<sizeof($vetor);$i++)
{
        // é impar
        if($vetor[$i]%2 != 0)
        {
                continue;
        }
        echo "O número " . $vetor[$i] . " é par.<br>";
}
```

- A função **sizeof** retorna o número de elementos do array

Dúvidas

INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
RIO GRANDE DO NORTE

Referências

- PHP Manual:
  - http://www.php.net/manual/pt_BR/index.php
- Desenvolvendo Websites com PHP
  - De Juliano Niederauer