1 Curso POO PHP Estruturas de control

1.1 Sumario

1.2 Estruturas de control

En PHP os guións constrúense sobre a base de sentenzas. Utilizando chaves, podes agrupar as sentenzas en conxuntos, que se comportan coma se fosen unha única sentenza.

Para definir o fluxo dun programa en PHP, ao igual que na maioría de linguaxes de programación, hai sentenzas para dous tipos de estruturas de control: sentenzas condicionais, que permiten definir as condicións baixo as que debe executarse unha sentenza ou un bloque de sentenzas; e sentenzas de bucle, coas que podes definir se unha sentenza ou conxunto de sentenzas se repite ou non, e baixo que condicións.

Ademais, en PHP podes usar tamén (aínda que non é recomendable) a sentenza **goto**, que che permite saltar directamente a outro punto do programa que indiques mediante unha etiqueta.

```
<?php
goto fin;
echo 'Non se executa';

fin:
echo 'Chegamos ao final';
?>
```

1.2.1 Estructuras condicionais

A sentenza **if** permite definir unha expresión para executar ou non a sentenza ou conxunto de sentenzas seguinte. Se a expresión se avalía a true (verdadeiro), a sentenza execútase. Se se avalía false (falso), non se executará.

Cando o resultado da expresión sexa false, podes utilizar **else** para indicar unha sentenza ou grupo de sentenzas a executar nese caso. Outra alternativa a else é utilizar **elseif** e escribir unha nova expresión que comezará un novo condicional.

```
<?php
if ($a > 0) {
   echo "\$a é positivo";
   echo "\$a ten o valor $a";
} elseif ($a == 0) {
   echo "\$a é negativo";
   echo "\$a ten o valor $a";
} else
   echo "\$a é igual a 0";
?>
```

Cando a sentenza if, elseif ou else actúe sobre unha única sentenza, non será necesario usar chaves. Terás que usar chaves para formar un conxunto de sentenzas sempre que queiras que o condicional actúe sobre máis dunha sentenza.

A sentenza **switch** é similar a enlazar varias sentenzas if comparando unha mesma variable con diferentes valores. Cada valor vai nunha sentenza **case**. Cando se encontra unha coincidencia, comezan a executarse as sentenzas seguintes ata que remata o bloque switch, ou ata que se encontra unha sentenza **break**. Se non existe coincidencia co valor de ningún case, execútanse as sentenzas do bloque **default**, en caso de que exista.

```
<?php
switch ($a) {
    case 0:
        echo "\$a é igual a 0";
        break;
    case 1:
    case 3:
```

```
    case 5:
    case 7:
    case 9:
        echo "\$a é impar positivo menor de 10";
        break;
    default:
        echo "\$a ten o valor $a";
}
?>
```

Tamén temos a posibilidade de crear expresións condicionais sinxelas empregando o operador ternario **?**.

```
$resultado = expr1 ? expr2 : expr3;
```

Cando expr1 é igual a *true*, o resultado é igual a expr2; en caso contrario o resultado é igual a expr3. Por exemplo:

```
$resultado = $divisor == 0 ? "NaN" : $dividendo / $divisor;
```

Dende PHP 5.3 é posible omitir o segundo operando. Neste caso, cando expr1 é igual a *true*, o resultado é igual a expr1. Emprégase principalmente para comprobar se un valor é nulo ou vacío, e ofrecer un valor alternativo. Por exemplo:

```
$cidade = $_REQUEST["cidade"] ?: "Santiago de Compostela";
```

1.2.2 Bucles

Empregando **while** podes definir un bucle que se executa mentres se cumpra unha expresión. A expresión avalíase antes de comezar cada execución do bucle.

```
<?php
$a=0;
while ($a<10) {
    $a++;
    echo "\$a ten o valor $a";
}
?>
```

do / **while** é un bucle similar ao anterior, pero a expresión avalíase ao final, co cal se asegura que a sentenza ou conxunto de sentenzas do bucle se executan polo menos unha vez.

```
<?php
$a=0;
do {
    echo "\$a ten o valor $a";
    $a++;
} while ($a<10)
?>
```

Os bucles **for** son os máis complexos de PHP. Ao igual que os da linguaxe C, compóñense de tres expresións:

for (expr1; expr2; expr3) sentenza ou conxunto de sentenzas;

- A primeira expresión, expr1, execútase só unha vez ao comezo do bucle.
- A segunda expresión, expr2, avalíase para saber se se debe executar ou non a sentenza ou conxunto de sentenzas. Se o resultado o false, o bucle remata.
- Se o resultado é true, execútanse as sentenzas e ao finalizar execútase a terceira expresión, expr3, e vólvese avaliar expr2 para decidir se se volve executar ou non o bucle.

```
<?php
for($a=0; $a<10; $a++)
    echo "\$a ten o valor $a";
?>
```

Podes aniñar calquera dos bucles anteriores en varios niveis. Tamén podes usar as sentenzas **break**, para saír do bucle, e **continue**, para omitir a execución das sentenzas restantes e volver á comprobación da expresión respectivamente.

1.2.3 Sintaxe alternativa das estruturas de control

Calquera das estruturas de control anteriores excepto **do-while** admite unha sintaxe alternativa, que consiste en cambiar a chave de apertura polo caracter dous puntos "**:**", e a chave de peche pola palabra clave **endif**, **endswitch**, **endwhile** ou **endfor** dependendo da estrutura que esteamos a empregar.

```
<?php
$a=0;
$a=0;
while ($a<10):
    $a++;
    echo "\$a ten o valor $a";
endwhile;
?>
```

1.2.4 Finalizar a execución

En PHP temos unha construcción da linguaxe, **exit** (ou o seu alias **die**), que se emprega cando se produce un erro grave para finalizar a execución do script actual.

Pode pasárselle como parámetro o valor de saída do script (un número enteiro) ou a mensaxe que queremos que se amose antes de rematar (unha cadea de texto). Se non leva parámetro, non é necesarios empregar parénteses na chamada.

--Víctor Lourido 14:43 25 jun 2013 (CEST)