Entorns de Desenvolupament

1. Variables
2. Instrucció echo
3. Estructures condicionals

Unió Europea
Fons social europeu
L'FSE inverteix en el teu futur

IES DR. LLUÍS SIMARRO

1. Variables en PHP

Les variables en PHP són representades amb un signe de dòlar (**$**) seguit pel nom de la variable. El nom de la variable és sensible a minúscules i majúscules. És a dir, les següents declaracions de variables són diferents entre elles:

$variable, $Variable, $variAble, $VariAble, etc. representen a diferents variables.

Una variable és un espai de la memòria de l'ordinador (en aquest cas del servidor) a la qual assignem un contingut que a grans trets pot ser un valor numèric (només números, amb el seu valor de càlcul) o alfanumèric (només text o text amb números, sense valor de càlcul).

Per declarar un nom de variable vàlid, aquest ha de començar amb una lletra (o si no comença amb una lletra, ho haurà de fer amb un caràcter de subratllat), seguit de qualsevol nombre de lletres, números i caràcters de subratllat, sense espais. Vegem alguns exemples:

Possible nom de variable	Validesa
$4variable	**Nom de variable invàlid.**
$_4variable	**Nom de variable vàlid.**
$Variable4	**Nom de variable vàlid.**
$Altra	**Nom de variable vàlid.**
$1_altra	**Nom de variable invàlid.**
$Variable_de_nom_molt_llarg	**Nom de variable vàlid.**
$ABC	**Nom de variable vàlid.**
$A B C	**Nom de variable invàlid.**
$A_y_B_x_C	**Nom de variable vàlid.**

Quan l'intèrpret PHP detecta una errada en la sintaxi en declarar una variable, quan tractem d'accedir a la pàgina php ens apareixerà un missatge d'error, si s'ha configurat correctament el fitxer **php.ini** escaient. Per exemple si en un arxiu php incloem una línia

com **$A B C = 5;** on el nom de variable és invàlid ja que té espais intermedis, ens apareixerà un missatge d'error:

Parse error: syntax error, unexpected identifier "B" in /var/www/html/entorns/index.php on line 20

Aquest missatge ens informa que hi ha un problema a l'arxiu php i ens indica la línia on l'intèrpret considera que està el problema.

Diferències entre els operadors d'assignació i igualtat.

Assignació (=)

L'operador d'assignació és "**=**". Es podria pensar que és com un "igual a". No ho és. L'operador igual en programació no té el mateix sentit que en matemàtiques. En programació, vol dir que el valor de l'expressió de la dreta s'estableix a l'operand de l'esquerra. Fixa't en aquest exemple i en els comentaris.

```
<?php
    $A = 3; // Assignem el valor 3 a la variable $A
    $B = "cadena"; // Assignem el valor "cadena" a la variable $B
?>
```

Operador d'igualtat (==)

L'operador de comparació d'igualtat és **==**. Com el seu propi nom indica, serveix per comparar dos valors o variables i no per assignar valors. Aquest operador retorna el resultat de l'operació lògica 'igual a' (que és un **booleà** s'avalua a `true` o `false`) tal com podem apreciar en el següent exemple:

```
<?php
        $A = 3;
        $B = 3;
        $A == $B; // El resultat de la comparació és TRUE perquè $A és igual a $A
        $A = 3;
        $B = 4;
        $A == $B; // El resultat de la comparació és FALSE perquè $A no és igual a $B
?>
```

Tipus de dades

En PHP no és obligatori indicar el tipus de dades a què pertany una variable com en altres llenguatges, sinó que els tipus de dades són establerts directament per l'intèrpret PHP, que és l'encarregat d'interpretar el codi. Ara veurem en una taula els diferents tipus de dades que es poden usar en PHP.

Tipus de dades	Definició
integer	• Els integers, enters o sencers, poden tenir diferents valors numèrics sencers que s'expressen amb diferents notacions. ➔ **$Variable = 18;** // Nombre enter positiu ➔ **$Variable = -18;** // Nombre enter negatiu ➔ **$Variable = 0x12;** // Notació hexadecimal, en principi no la farem servir
float o double	• Aquest tipus de dades són els números de punt flotant als que normalment anomenem "nombres decimals", per exemple, 9.876. Tots dos tenen molta precisió, però double és el més precís (amb més decimals). La sintaxi per utilitzar-los és bastant simple: ➔ **$Variable = 9.876;**
string	• El tipus de dades string, també conegut com a cadena de caràcters, s'expressa amb la següent sintaxi: ➔ **$Variable = "Jo sóc una cadena";**

boolean	• Es tracta d'un tipus lògic. Els seus possibles valors són true (vertader) o false (fals). ➔ **$Variable = true;** ➔ **$Variable = false;**

2. Instrucció echo

Aquesta instrucció moltes vegades es diu que serveix perquè es mostre per pantalla un text, una variable, o una combinació de text i variables. Però realment no és així (encara que podem usar-la per això), sinó que serveix per a inserir text dins del document HTML subjacent al codi PHP.

Per exemple la instrucció

```
echo "eixida per pantalla.";
```

farà que s'inserisca aquest text en el document html. Si ho fem entre les etiquetes **<body>** i **</body>** ho veurem per pantalla perquè qualsevol text inserit en aquest lloc es mostrarà per pantalla, no perquè la instrucció echo done lloc al fet que es mostre per pantalla.

No obstant això

```
echo "<h1>Eixida per pantalla. </h1>";
```

no farà que es mostren per pantalla les etiquetes h1 i el text, sinó que introduirà en el document HTML les etiquetes amb el text, aquestes etiquetes funcionaran com a codi HTML i el que es mostrarà per pantalla serà **Eixida per pantalla** amb format de títol **h1** (lletres més grans del normal).

En general en php són vàlides tant les cometes dobles com les cometes simples, de manera que es pot escriure tant

```
echo "<h1> Eixida per pantalla. </h1>";
```

com

```
echo '<h1>Eixida per pantalla. </h1> ';
```

Els parèntesis no són necessaris (echo en realitat no és una funció sinó que és el que es denomina una **"construcció del llenguatge"**, per aquest motiu no són estrictament necessaris els parèntesis), però es poden incloure si es desitja. Exemple:

```
echo ('amb parèntesis i cometes simples');
```

La diferència fonamental entre utilitzar cometes simples i dobles és que amb aquestes últimes es permet introduir variables PHP definides prèviament que seran substituïdes pel seu valor corresponent. El següent exemple substituirà el valor de la variable **$cadena** pel que s'ha indicat:

```
<?php
        $cadena = "Hola";
        echo "$cadena món!";
?>
```

El que veuríem al navegador seria:

Hola món!

En canvi, si utilitzàrem les cometes simples, l'expressió **$cadena** s'entendria com un literal (text, no una variable), i el valor no seria substituït.

```
<?php
        $cadena = "Hola";
        echo '$cadena món!';
?>
```

El que veuríem al navegador seria:

$cadena món!

Si crearem un arxiu php amb el següent codi:

```
<?php
        echo "eixida per pantalla.";
        $A = 5;
```

```
        echo $A;
        echo "El valor de la variable A és $A.";
?>
```

La visualització al navegador seria similar a aquesta:

eixida per pantalla. 5 El valor de la variable A és 5.

Hem de tenir en compte que $A es mostra per pantalla com a 5 perquè l'intèrpret de PHP considera que ha de mostrar sempre el valor de la variable si utilitzem cometes dobles, **fins i tot si va dins d'un text**. Si canviarem l'última línia anterior per aquesta:

```
echo "El valor de la variable \$A és $A.";
```

Comprovaríem que es mostra per pantalla:

eixida per pantalla. 5 El valor de la variable $A és 5.

Quan fiquem abans del nom d'una variable el **caràcter ** fem que s'interprete com a text en lloc de com una variable. El caràcter \ se sol anomenar "**caràcter d'escapament**" perquè permet que es considere text alguna cosa que en principi s'anava a interpretar com una altra cosa (com una variable per exemple).

Si creem un fitxer php amb el següent codi i el visualitzem:

```
<html>
<head>
<title> Exemple echo </ title>
<meta charset = "utf-8"/>
</head>
<body>
<?php
        echo 'amb cometes simples <br/>';
        echo ('amb parèntesis i cometes simples <br/>');
```

```
        echo ("amb parèntesis i cometes dobles <br/>");
        echo '<h1> Eixida per pantalla </h1> <br/>';
        echo "eixida per pantalla.";
        $A = 5;
        echo $A;
        echo "El valor de la variable A és $A.";
?>
</body>
</html>
```

La visualització al navegador haurà de ser semblant a:

amb cometes simples
amb parèntesis i cometes simples
amb parèntesis i cometes dobles
Eixida per pantalla
eixida per pantalla. 5 El valor de la variable A es 5.

L'etiqueta
 no es mostra per pantalla perquè és codi HTML, i el navegador quan detecta aquest codi HTML introdueix un salt de línia en lloc de mostrar-ho per pantalla. Per comprendre bé la utilitat de la instrucció echo pensem en aquesta equivalència:

`<body>`
`<? php`
`echo '<h1> Hola </h1>'`
`?>`
`</body>`
	Equival a	`<body>`
		`<h1> Hola </h1>`
		`</body>`

El navegador rep l'HTML "**equivalent**" generat i l'interpreta, donant lloc als resultats que es mostraran per pantalla.

3. Estructures condicionals

En PHP hi ha una sèrie d'instruccions que permeten prendre **decisions lògiques** quan programem: aquestes instruccions, comuns a diferents llenguatges de programació, se solen anomenar de forma general "condicionals".

- **if ... else**
- **if ... elseif ... else**
- **switch**

Condicional if ... else

Aquestes instruccions són potser les més usades de tots els llenguatges de programació, ja que són les més bàsiques. **if** (traduït: si ...) **else** (traduït: sinó ...). L'estructura general d'un condicional **if else** és la següent:

```
if (expressió) {
        Sentència1; Sentència2; Sentència3; ...;
} else {
        Sentència A; Sentència B; Sentència C; ...;
}
```

En aquest tros de codi estaríem dient que si l'expressió es compleix s'executen les sentències 1, 2, 3 ..., i en cas que no es complira l'expressió s'executarien les sentències que estan dins de l'**else**, és a dir, les sentències A , B, C ...

Veiem aquest exemple:

```
<?php
        $galetes = true;
        if($galetes == true) {
                echo 'Hi ha galetes';
        } else {
```

```
            echo 'No hi ha galetes';
        }
?>
```

En aquest exemple, estem donant un valor a $galetes i comprovem si el valor d'aquesta variable és **true** o no, introduint-lo en l'expressió del condicional. Tal qual està, aquest exemple produiria el resultat '**Hi ha galetes**', tot i que si canviàrem **$galetes = true;** per **$galetes = false;** donaria com a resultat la segona sentència.

Modifiquem el codi anterior:

```
<?php
        $galetes = true;
        if ($galetes == true) {
?>
<br/>
<hr/>
<?php
        echo 'Hi ha galetes';
        echo '<br/>';
        echo '<br/>';
        echo '<hr/>';
        } else {
            echo 'No hi ha galetes';
        }
?>
```

Observem que el codi produeix en essència el mateix resultat. Únicament hem introduït alguns caràcters per millorar la forma en què es veu. Hem escrit un **fragment** de codi **php**, després un **fragment** de codi **html** i després un altre **fragment** en **php**. No hem utilitzat les etiquetes d'obertura i tancament d'html, encara que haguera estat potser més correcte fer-ho, és a dir, escriure:

```
<html>
<?php
        $galetes = true;
        if ($galetes == true) {
?>
<br/>
<hr/>
<?php
        echo 'Hi ha galetes';
        echo '<br/>';
        echo '<br/>';
        echo '<hr/>';
        } else {
                echo 'No hi ha galetes';
        }
?>
</html>
```

Un salt de línia, que introduïm gràcies a l'etiqueta **
, podem escriure'l tant fora del codi php com dins d'ell. Per introduir-lo dins del PHP, hem de posar-lo dins d'un **echo, que és la forma que tenim de indicar-li a php que done eixida a aquesta etiqueta com a codi html.

Els navegadors són capaços d'interpretar el codi encara que falten algunes etiquetes o continguen alguns errors. Aquesta és una de les qüestions que diferencia HTML de llenguatges de programació en sentit estricte com C, Java o Visual Basic, on la sintaxi és molt més estricta.

Si introduïm una etiqueta html com **
** directament dins el codi php, el navegador es mostra un missatge d'error d'aquest tipus:

Par-se error: syntax error, unexpected '<' in /localhost/prova.php on line 11

Aquesta és una qüestió que s'ha de tenir en compte: per **introduir html** has de **tancar un bloc php** o utilitzar una **instrucció echo dins del bloc php**. La instrucció **echo** significa "donar eixida html al que va a continuació".

PHP ofereix una sintaxi alternativa per a algunes de les seues estructures de control; per exemple **if**, **while**, **for**, **foreach** i **switch**. En el cas de l'**if**, la forma bàsica de la sintaxi alternativa consisteix a canviar la clau d'obertura a dos punts (**:**) i la clau de tancament a **endif ;**. Per exemple

```
<?php
        $a = 5;
        if ($a == 5):
?>
        <p>A és igual 5</p>
<?php
        endif;
?>
```

Exemple amb sintaxi alternativa

```
<?php
        $a = 5;
        if ($a == 5){
?>
        <p>A és igual 5</p>
<?php
        }
?>
```

Exemple amb sintaxi tradicional

I amb **else**:

```
<?php
        $a = 4;
        if ($a == 5):
?>
        <p>A és igual 5</p>
<?php
        else:
?>
        <p>A no és igual a 5</p>
<?php
        endif;
?>
```

Exemple amb sintaxi alternativa

```
<?php
        $a = 4;
        if ($a == 5){
?>
        <p>A és igual 5</p>
<?php
        } else {
?>
        <p>A no és igual a 5</p>
<?php
        }
?>
```

Exemple amb sintaxi tradicional

Condicionals if ... else if ... else

L'estructura d'aquestes instruccions ve a ser molt similar a l'anterior. El seu significat: "**Si passa això, fes això, sinó, si passa aquesta altra cosa, fes això altre, sinó si passa aquesta altra cosa, fes ...**". L'estructura general d'un condicional **if else if [else]** és:

```
if (expressió) {
        Sentència1; Sentència2; Sentència3, ...;
} else if (expressió) {
        SentènciaA; SentènciaB; SentènciaC; ...;
} else {
        SentènciaM; SentènciaN; SentènciaO; ...;
}
```

Aquesta és la sintaxi amb la qual s'hauria d'utilitzar **else if**. Un **else if** s'executarà, sempre que no hi haja alguna resposta veritable (**true**) a l'**if** o **else if** anterior. Si s'arriba al final dels **else if**, i cap ha estat veritable (**true**), s'executarà el que està dins el **else** final.

Poden haver diversos **else if**, però aquests estan sempre dins d'un **if** i acabaran amb un **else** (només un) final.

Veiem el següent exemple:

```
<?php
        $test = 33; // és més gran que 40 o 35 o 30?// és més gran que 40 o 35 o 30?
        if ($test> 40) {
                echo "Sí, $test és més gran que 40.";
        } else if ($test> 35) {
                echo "Sí, $test és més gran que 35.";
        } else if ($test> 30) {
                echo "Sí, $test és més gran que 30.";
```

```
    } else {
        echo "No, $test és menor que 40, 35 i 30.";
    }
?>
```

En aquest cas la resposta seria: **Sí, 33 és més gran que 30.** Si el valor de **$test** es canvia a 22, la resposta seria: **No, 22 és menor que 40, 35 i 30.**

També es pot utilitzar la sintaxi alternativa en aquest cas. Per exemple:

```
<?php
  $a = 7;
  if ($a == 5):
    echo "<h3>a val 5</h3>";
  elseif ($a == 6): //junt
    echo "<h3>a val 6</h3>";
  else:
    echo "<h3>a no val ni 6 ni 7</h3>";
  endif;
?>
```

Exemple amb sintaxi alternativa

```
<?php
  $a = 7;
  if ($a == 5){
    echo "<h3>a val 5</h3>";
  }else if ($a == 6){ //separat
    echo "<h3>a val 6</h3>";
  }else{
    echo "<h3>a no val ni 6 ni 7</h3>";
  }
?>
```

Exemple amb sintaxi tradicional

Operador condicional ternari

Per escriure condicionals l'estructura bàsica de llenguatges com PHP, JavaScript, Java, i altres és la sentència **if**. Un altre tipus de condicionals com el **switch** o el **else if** poden ser reemplaçats per **if**. Hi ha una altra sintaxi de condicional que s'anomena operador condicional ternari i que s'escriu amb expressions que inclouen una interrogació i dos punts com: **a? b: c;**

La sintaxi i significat de l'operador condicional ternari (vàlida per a diferents llenguatges, com PHP, JavaScript, Java, etc.) és la següent:

expressióAmbValorBooleà? Expressió1: Expressió2;

Interpretació: **si l'expresióAmbValorBooleà és certa s'executa l'Expressió1, i**

en cas contrari s'executa l'Expressió2.

Exemples. Suposem que tenim quatre variables: $A = 5, $B = 3, $C = -7 i $D = 5

expressió	resultat
$A == $5? dispara (): espera ();	S'executa dispara ()
$A <$B? dispara (): espera ();	S'executa espera ()
$B <$C? dispara (): espera ();	S'executa espera ()
$A <$B && $B> $C? dispara (): espera ();	S'executa espera ()
($A <$B && $B> $C) || $B == 3? dispara (): espera ();	S'executa dispara ()
$A == 5? $A = 20: $A = 1;	Si A valia 5 ara val 20, cas contrari ara val 1.

L'operador ternari pot ser inserit en sentències d'execució on no es permet la inserció d'**if**, per exemple en una operació d'assignació o en la variable de control d'un bucle **for**. Utilitza l'operador condicional ternari té avantatges i inconvenients.

Avantatges i inconvenients de l'operador condicional ternari

A la següent taula resumim els avantatges i inconvenients de l'operador condicional ternari comparant-lo amb l'**if** tradicional.

Condicional ternari	if tradicional
Permet l'escriptura compacta, permetent estalviar escriptura de codi.	**Obliga a escriure més sentències per aconseguir el mateix resultat.**
Resulta més difícil de llegir, entendre i depurar	**Resulta més fàcil de llegir, entendre i depurar**
No tots els programadors el fan servir, alguns ni tan sols el coneixen.	**Tots els programadors el fan servir i el coneixen.**
S'admet en la sintaxi dels llenguatges en llocs on no s'admet la sentència if	**No és vàlid en certes ubicacions on només s'admeten expressions, però pot fer-se l'avaluació abans del punt on siga necessari el condicional.**
Criticat per alguns experts, adorat per	**Ús i acceptació quasi-universal**

altres	

Exemples d'ús en php

```
<!DOCTYPE html>
<html>
<head>
<title> Exemple operador ternari</title>
<meta charset = "utf-8"/>
</head>
<body>
<div id = "cap">
<h2> Exemple Operador Ternari </h2>
<h3> Exemples PHP </h3>
</div>
<div>
<p>
<?php
        $A = 5; $B = 3; $C = -7; $D = 5;
        $msg = '';
        $A == 5? $msg = 'A és 5 <br/>': $msg = 'A no és 5 <br/>';
        echo $msg;
        $A <$B? $msg = 'A menor que B <br/>': $msg = 'A major o igual que B <br/>';
        echo $msg;
        $B <$C? $msg = 'B menor que C <br/>': $msg = 'B major o igual que C <br/>';
        echo $msg;
?>
</p>
</div>
</body>
</html>
```

El resultat esperat és que es mostre per pantalla:

Exemple Operador Ternari

exemples PHP

A és 5

A major o igual que B

B major o igual que C