

El operador if

Operadores condicionales

Este tipo de operadores son el auténtico *cerebro* de cualquier aplicación que desarrollemos en PHP o en cualquier otro lenguaje de programación.

Los operadores condicionales son la herramienta que permite tomar decisiones tales como: *hacer o no hacer*, y también: *hacer algo bajo determinadas condiciones* y otra *cosa distinta* en caso de que *no se cumplan*.

Condiciones

Aunque para simplificar los ejemplos vamos a utilizar en ellos una sola condición, este operador permite incluir como tal cualquier estructura lógica, del tipo que hemos visto en la página anterior, por compleja que esta sea.

Alternativas de sintaxis

Como iremos viendo a lo largo de estas líneas, este operador permite diferentes formas de sintaxis que podemos utilizar según nuestra conveniencia.

La forma más simple es:

```
if(condición)
    ..instrucción...;
```

Si se cumple la *condición* establecida en el paréntesis se ejecutará la primera instrucción que se incluya a continuación de ella.

Cualquier otra instrucción que hubiera a continuación de esa primera no estaría afectada por el condicional y se ejecutaría en cualquier circunstancia.

Observa que, aunque hemos puesto *if(condición)* en una línea independiente, *no lleva punto y coma detrás*.

```
if(condición){
    ..instrucción 1...;
    ..instrucción 2...;
    ....;
}
```

Es una ampliación del caso anterior. Cuando es necesario que –en caso de que se cumpla la condición o condiciones– se ejecute más de una instrucción, se añade una { para indicar que habrá varias instrucciones, se escriben estas y

```
<?
# Definamos dos variables y lasignémosles valores.
# Hubieran podido obtenerse por cualquier otro procedimiento:
# desde un array,
# a través de un formulario cuya acción ejecute este script, etc.

$A=3; $B="3";
if ($A==$B)
    print ("A es igual B");

# cualquier otra instrucción que incluyéramos de aquí
# en adelante se ejecutaría independientemente de que la condición
# se cumpla ó no ya que esta forma de if (sin llaves)
# únicamente considera la primera instrucción
# comprobémoslo en este otro supuesto

if ($A<$B)
    print ("A es menor que B");
    print("<br>A no es menor que b, pero esto saldrá<br>");
    print("Esta es la segunda instrucción. No la condicionaré el if");
?>
```

ejemplo24.php

```
<?
$A=3; $B="3";
# en este caso cerraremos entre llaves las líneas
# que deben ejecutarse si se cumple la condición

if ($A==$B){
    print ("A es igual B");
    echo "<br>";
    echo "Este if tiene varias instrucciones contenidas entre llaves";
}

# una sintaxis alternativa a las llaves
# sustituyamos la { por : y la } por endif
if ($A==$B):
    print ("A es igual B");
    echo "<br>";
    echo "Hemos cambiado {} por : endif";
endif;

?>
```

ejemplo25.php

```
<?
$a=5;
# observa que ponemos la etiqueta de fin de script
# después de la llave de apertura
if ($a==5){ ?>
<!-- Aquí estamos poniendo HTML puro
no estamos dentro del script PHP //-->
<H1>Esto no ha sido interpretado por PHP</H1>

<!-- en la línea siguiente a este comentario
volveremos a PHP para insertar la llave que indica el fin del if //-->
<? } ?>

<?
# hagamos lo mismo cambiando {} por : endif
```

mediante } se señala el final.

```
if(condición) :  
    ..instrucción 1... ;  
    ..instrucción 2... ;  
    .... ;  
endif;
```

Esta otra forma del condicional se comporta como la anterior pero con otra sintaxis. Se *sustituye* la { de apertura por : y la } de cierre por **endif**

```
if(condicion){ ?>  
    ..Etiquetas HTML... ;  
    ..HTML... ;  
    .... ;  
<? ?>
```

PHP permite la utilización del operador condicional **if** con esta sintaxis. Una primer *script* PHP establece la condición. Todo lo contenido entre ese primer script y el de cierre: `<? ?>` será código HTML (está fuera del script), que se insertará en el documento sólo en el caso de que se cumpla la condición.

```
if(condicion) : ?>  
    ..Etiquetas HTML... ;  
    ..HTML... ;  
    .... ;  
<? endif; ?>
```

Idéntica a la anterior, con la sintaxis : **endif**.

If ... else

El operador condicional tiene una interesante ampliación. En conjunción con **else** permite añadir instrucciones que sólo serían ejecutadas en caso de no cumplirse la condición.

Esta nueva opción se habilita mediante la siguiente sintaxis:

```
if(condicion){  
    ... instrucciones...  
    ... a ejecutar cuando  
    se cumple la condición  
    } else {  
    ... instrucciones...  
    ... a ejecutar cuando NO  
    se cumple la condición  
    }
```

permitiendo también la sintaxis alternativa :, **endif**, aunque en este caso hay que hacer una precisión -puedes verla aquí debajo- la llave de cierre que iba delante de **else** se elimina y no es sustituida por ningún carácter ni símbolo especial.

```
<?if(condicion): ?>
```

```
... código HTML  
... a ejecutar cuando  
se cumple la condición
```

```
<? else: ?>
```

```
-- ,-- ,-- ,--  
<!-- Aquí estamos poniendo HTML puro  
no estamos dentro del script PHP //-->  
<H2>Esto tampoco sido interpretado por PHP</H2>  
  
<!-- en la línea siguiente a este comentario  
volveremos a PHP para insertar la llave que indica el fin del if //-->  
<? endif; ?>
```

ejemplo26.php

La estructura if ... else

```
<?  
$A=3; $B="4";  
if ($A==$B){  
#estas instrucciones se ejecutarían si se cumple la condición  
    print ("A es igual B");  
    echo "<br>";  
    echo "Este if tiene varias intrucciones";  
    }else{  
# estas se ejecutarían en el caso de no cumplirse  
# las condiciones especificadas en el fi  
    print("A no es igual que B");  
    echo "<br>";  
    echo ("La estructura de control se ha desviado al else");  
}  
>
```

ejemplo27.php

```
<?  
$a=3;  
# observa que ponemos la etiqueta de fin de script  
# después de los dos puntos  
if ($a==5): ?>  
<!-- Aquí estamos poniendo HTML puro  
no estamos dentro del script PHP //-->  
<H1>Esto no es PHP. A es igual 5</H1>  
  
<!-- en la línea siguiente a este comentario  
volveremos a PHP para insertar el else seguido de dos puntos  
y cerramos de nuevo el script con ?>!-->  
<? else: ?>  
  
<!-- Aquí más HTML para el (else)  
caso de que no se cumpla la condición //-->  
<H2>Esto no es PHP. Es el resultado del ELSE</H2>  
  
<!--  
volveremos a PHP para insertar en endif que indica el fin del if //-->  
<? endif; ?>
```

ejemplo28.php

El operador condicional ternario

```
<? $a=5;  
($a==8) ? ($B="El valor de a es 8") : ($B="El valor de a no es 8");  
echo $B;  
>
```

Ejemplo con a=8

Ejemplo con a=5

La estructura if ... elseif... else

... código HTML...

... a ejecutar cuando NO se cumple la condición

<? endif; ?>

En algunos casos resulta útil y cómodo el uso de esta otra posibilidad de sintaxis:

(condición) ? (opc1) : (opc2)

Si se cumple la **condición** se ejecuta la **opc1**, pero en el caso de que no se cumpla se ejecutará la **opc2**.

If ... elseif .. else

Otra posibilidad dentro de la estructura de los operadores condicionales es la inclusión de **elseif**.

Esta es la sintaxis. (Dentro de ella tienes los comentarios explicativos).

if(condición1){

... instrucciones...

... a ejecutar cuando

se cumple la condición1

}elseif(condición2){

... instrucciones...

... a ejecutar cuando

se cumple la condición2

sin cumplirse condición1

} else {

... instrucciones...

... a ejecutar cuando NO

se cumple ni la condición1

ni la condición2

}

Condicionales anidados

El *anidado* no es otra cosa que el equivalente a los *paréntesis dentro de paréntesis* en las matemáticas. Y este operador lo permite, con una única condición, que verás en esta muestra de sintaxis.

if(condición1){

... instrucciones...

if(condición2){

... instrucciones...

} else {

...instrucciones

}

}elseif{

... instrucciones...

...instrucciones...

}

Observa que todo el bloque if.. else... marcado en azul se cierra antes de abrir la opción else marcada en marrón. Es **obligatorio** que así sea. De igual forma, podríamos insertar bloques sucesivos hasta llenar a

```
<? $a=1;
  if ($a==1){
    echo "El valor de la variable A es 1";
  }elseif ($a==2){
    echo "El valor de la variable A es 2";
  }elseif ($a==3){
    echo "El valor de la variable A es 3";
  }else{
    echo "La variable A no es 1, ni 2, ni 3";
  }
?>
```

Ejemplo con a=3

Ejemplo con a=-7

```
<? $a=1;
  if ($a==1): ?>
    <H1>A es igual a 1</H1>
  <? elseif($a==2): ?>
    <H1>A es igual a 2</H1>
  <? elseif($a==3): ?>
    <H1>A es igual a 3</H1>
  <? else: ?>
    <H1>A no es igual ni a 1, ni a 2, ni a 3</H1>
<? endif;
?>
```

Ejemplo con a=2

Ejemplo con a=8

Ejercicio nº 19

Diseña un formulario –**ejercicio19a.php**– con un input tipo texto en el que puedas escribir números. Al pulsar el botón de enviar debe llamar a un script –**ejercicio19b.php**– que debe decirnos si el número enviado fue: **positivo, cero o negativo**.

A la página **ejercicio19b.php** añádele un *enlace* HTML que permita volver a la página anterior.

Ejercicio nº 20

En el ejercicio nº 10 –**puedes verlo pulsando aquí**– diseñaste un cuestionario en el que formulabas dos preguntas. Utilizando un formulario similar, pero únicamente con la primera pregunta –**puedes modificarlo y guardarlo como ejercicio20a.php**– debes crear un script de modo que al recibir el formulario muestre en pantalla «Respuesta correcta» ó «Respuesta incorrecta».

Como es lógico, en ese script –**puedes llamarlo ejercicio20b.php**– debes incluir en una variable el valor de la respuesta correcta y compararla con la recibida a través del formulario.

Ejercicio nº 21

Amplía el ejercicio anterior a las dos preguntas que se formulaban en el nº10. Ahora deberíamos saber si ha sido correcta o no la respuesta a cada una de las preguntas. Puedes llamar **ejercicio21a.php** y **ejercicio21b.php** a los documentos que crees para este ejercicio.

Restringir accesos

Las variables predefinidas **\$_SERVER['HTTP_REFERER']** (en el caso de PHP 4.1.0 o superior) y

crear una estructura tan amplia como fuera necesaria.

Como ves, todo un mundo de posibilidades.

Una aplicación a la seguridad

En páginas anteriores hemos hecho algunas alusiones a la seguridad.

Decíamos que los envíos de información por medio de los formularios no eran seguros porque, dada la transparencia de su código, pueden ser reproducidos y utilizados desde cualquier otro sitio distinto a nuestro *servidor*.

Una sencillo condicional puede resolver ese problema. Lo puedes ver en el ejemplo que tienes aquí a la derecha.

La función `exit()`

PHP dispone de una función `exit()` muy útil a los efectos del comentario anterior.

Cuando se ejecuta `exit()` se **interrumpe** la ejecución del script con lo que la respuesta del servidor a la petición del **cliente** incluirá únicamente los contenidos generados antes de su ejecución.

`$HTTP_SERVER_VARS['HTTP_REFERER']` (en todos los casos) recogen la ruta completa de la página desde la que hemos accedido a la actual.

```
<?
# el condicional if estable como condición
# que el acceso a este script proceda de la dirección indicada
# en este caso hemos puesto como condición que ese valor
# sea la dirección de esta página
if($_SERVER['HTTP_REFERER']=="http://localhost/cursophp/php37.php"){
# si accedemos desde esta página, el enlace que tienes aquí debajo
# veremos que aparece este print, es decir se visualizaría todo
# lo contenido antes del else
    print "ejecuto sin problemas el script."
    print "Vienes de:".$_SERVER['HTTP_REFERER'];
}else{
# si accedes desde un sitio diferente te aparecerá este mensaje
# puedes probar escribiendo en tu navegador
# http://localhost/cursophp/ejemplo35.php
# y comprobarás que aparece este mensaje
"No puedes ver esta pagina";
    exit;
}
?>
```

ejemplo35.php

Anterior

Índice

Siguiente