Programación Web con PHP y MySQL– Nivel 2

Unidad 5: Estructuras de control

Contacto: consutlas@elearning-total.com
Web: www.elearning-total.com

Indice

Unidad 5: Estructuras de control

Estructuras de control	4
Condicionales	4
Else	5
Switch	9
Links con variables	13
Variable $_GET	17
Variable $_POST	19

Objetivos

Que el alumno logre:

- Aprender los conceptos de paso de valores entre archivos

Estructuras de Control

A partir de ahora vamos a dotar de más "**dinamismo**" a nuestros scripts ya que a partir de diversas estructuras indicaremos que acción debe realizar en cada caso.

La programación nos exige en muchas ocasiones la repetición de acciones sucesivas o la elección de una determinada secuencia y no de otra dependiendo de las condiciones específicas de la ejecución.

Este tipo de acciones pueden ser llevadas a cabo gracias a una paleta de instrucciones presentes en la mayoría de los lenguajes.

CONDICIONALES

Son estructuras que evaluando una condición dada, "deciden" si ejecutar cierto bloque de código u otro. Cuando queremos que el programa, llegado a un cierto punto, tome un camino concreto en determinados casos y otro diferente si las condiciones de ejecución difieren, nos servimos del conjunto de instrucciones ***if, else y elseif.***

If

Se trata de una estructura de control utilizada para tomar decisiones según se cumpla una condición (o varias) o no.

La construcción *if* es una de las más importantes características de muchos lenguajes, incluido PHP.

La estructura de base de este tipo de instrucciones es la siguiente:

```
if (condición)
{
Instrucción 1;
Instrucción 2;
...
```

```
}
```

Para decirlo de otro modo, la construcción *if* permite condicionar la ejecución de un bloque de sentencias al cumplimiento de una condición, es decir que si la condición es verdadera, se ejecutarán la Instrucción 1 y la Instrucción 2.

ELSE

A menudo queremos ejecutar una sentencia si se cumple una cierta condición, y una sentencia distinta si la condición no se cumple. Esto es para lo que sirve la sentencia *else*.

else extiende una sentencia *if* para ejecutar una sentencia en caso de que la expresión en la sentencia *if* se evalúe como FALSE.

```
if (condición)
{
Instrucción 1;
Instrucción 2;
}
else
{
Instrucción A;
Instrucción B;
}
```

Llegados a este punto, el programa verificará el cumplimiento o no de la condición. Si la condición es cierta las instrucciones 1 y 2 serán ejecutadas. De lo contrario (*else*), las instrucciones A y B serán llevadas a cabo.

Es decir:

```
if (condición) { //Código a ejecutar si la condición es verdadera
}
else { //Otro código a ejecutar en caso contrario
}
```

Ejemplo:

```
<?php
$x=4;
if ($x>5) {
echo "El valor de x es mayor a 5";
}
else {
echo "El valor de x es menor o igual a 5";
}
?>
```

Salida:

El valor de x es menor o igual a 5

Esta estructura de base puede complicarse un poco más si tenemos cuenta que no necesariamente todo es blanco o negro y que muchas posibilidades pueden darse.

Es por ello que otras condiciones pueden plantearse dentro de la condición principal.

Hablamos por lo tanto de condiciones anidadas que tendrían una estructura del siguiente tipo:

```
if (condición1) {
Instrucción 1;
Instrucción 2; ...
} else {
        if (condición2) {
        Instrucción A;
        Instrucción B;
        ... } else {
        Instrucción X ... }
}
```

De este modo podríamos introducir tantas condiciones como queramos dentro de una condición principal.

Veamos un ejemplo con la utilización de la función de PHP ***rand()***, que genera un número entero aleatorio entre dos valores. Lo primero que nos plantearemos es generar ese valor aleatorio (es decir lo elige la máquina al azar) comprendido entre 1 y 10.

Luego mostraremos un mensaje si es menor o igual a 5 o si es mayor a 5.

El programa completo sería:

```
<html>
<head>
<title>Problema</title>
</head>
<body>
        <?php
        $valor=rand(1,10);
        echo "<p>El valor sorteado es $valor</p>";
                if ($valor<=5)
                {
                echo "<p>Es menor o igual a 5</p>";
                }
                else
                {
                echo "<p>Es mayor a 5</p>";
                }
        ?>
</body>
</html>
```

Veamos un nuevo ejemplo, un poco más complejo, esta vez se deben cumplir dos condiciones para que el IF sea TRUE, para ello usamos el operador Lógico && que viéramos en la unidad anterior.

```
If (($edad>=18) && ($dinero>5)){
echo "<p>Puedes comprar cerveza porque tienes 18 y tu dinero es mayor que 5</p>";
}
else{
echo "<p>O no tienes más de 5 pesos o no tienes 18 años, aléjate de la cerveza! </p>" ;
}
```

Elseif

De gran ayuda es la instrucción elseif que permite en una sola línea introducir una condición adicional. Este tipo de instrucción simplifica ligeramente la sintaxis que acabamos de ver:

```
if (condición1)
{
Instrucción 1;
Instrucción 2;
...
}
elseif (condición2)
{
Instrucción A;
Instrucción B;
...
}
else
{
Instrucción X
...
}
```

Ejemplo:

```
<?php
$test = 33;
if ($test > 40) {
echo " $test es mayor que 40.";
}
elseif ($test > 35) {
echo " $test es mayor que 35.";
}
elseif ($test > 30) {
echo " $test es mayor que 30.";
}
```

```
else {
echo " $test es menor que 40, 35 y 30.";
}
?>
```

En este caso la respuesta sería:

```
33 es mayor que 30.
```

No está de más mencionar, que para plantear las distintas condiciones, debemos tener en cuenta que disponemos de los siguientes operadores:

= = para ver si una variable es igual a otra.
!= distinto.
>= mayor o igual.
> mayor.
<= menor o igual
< menor

Para comparar usamos dos signos igual (= =), ya que si utilizáramos uno solo estaríamos asignando no comparando.

SWITCH

Un switch es una estructura que evalúa el valor de una variable o expresión y depende de su valor, ejecuta cierto bloque de código.

Sintaxis:

```
switch (expresión) {
case "a": //Código a ejecutar;
break;
case "b": //Código a ejecutar;
break;
case "c": //Código a ejecutar;
break;
default: //Código a ejecutar por default.
}
```

Un switch busca dentro de los “case” el valor de la variable o expresión evaluada (generalmente se evalúan variables). Si lo encuentra, ejecuta el código correspondiente.

Si no lo encuentra, ejecuta el código por default.

Observaciones:
El case por default es opcional. Es importante poner “break; “ al final de cada bloque de código dentro de cada case para que el switch no siga comparando al valor de la variable con los case que le siguen al correcto.

Ejemplo:

```
<?php
$examen="Bueno";
switch($examen) {
case "Excelente": echo "Su nota fue 10. ";
break;
case "Muy bien": echo "Su nota fue 8 o 9. ";
break;
case "Bueno": echo "Su nota fue 6 o 7";
break;
case "Regular": echo "Su nota fue 4 o 5";
break;
default: echo "Su nota fue menor o igual a 3. ";
}
?>
```

Salida:

```
Su nota fue 6 o 7
```

Las instrucciones **If...Elseif...Else** permiten seleccionar entre más de dos posibles condiciones que puedan ser evaluadas como true o false, pero en ocasiones nos encontraremos con casos en que queremos evaluar condiciones con una estructura como esta:

```
If ($variable==Valor1)
{
... Sentencias;
}
```

```
elseif($variable==Valor2)
{
... Sentencias;
}
elseif($variable==Valor3)
{
... Sentencias;
}else
{
... Sentencias;
}
```

En las que esperamos realizar una acción determinada según el valor tomado por una variable. Para estos casos la mayoría de los lenguajes de programación incluyen una instrucción que permite "seleccionar" entre los posibles valores de una variable: la instrucción **switch** (Español: SELECCIONAR).

Estructura:

```
switch($variable)
{
case Valor1:
... Sentencias;
break;
case Valor2:
... Sentencias;
break;
case Valor3:
... Sentencias;
break;
default:
... Sentencias;
}
```

En esta estructura, $variable es la variable que será comparada sucesivamente con cada una de las instrucciones **case** hasta hallar la que corresponda al valor de $variable, una vez que la encuentre se ejecutarán sucesivamente todas las instrucciones pertenecientes al **switch** hasta hallar la siguiente instrucción **break**; esta última hace un salto pasando el control del programa a la instrucción inmediata siguiente fuera del **switch**.

Si ninguna de las instrucciones **case** corresponde al valor de $variable se ejecutarán solo las instrucciones siguientes al **default**. Al igual que **else**, la instrucción **default** es opcional.

Links con variables

Vamos a ver a continuación como podemos pasar datos, a través de variables, entre distintas páginas de nuestro sitio.

La primera de las formas de "definirle un valor" o "setear" una variable: es a través los links. Cuando en HTML hacemos un link, habitualmente se parece a esto:

```
<html>
<head>
<title>Paso de variables por PHP</title>
</head>
<body>
<a href="destino.html">Esto es un link</a>
</body>
</html>
```

El efecto del tag <a> ("anchor", o "ancla", que nos enlaza a otra página) es pedirle al servidor que nos "muestre" o nos entregue la página especificada en el atributo "href" (es la abreviatura de "hiper-referencia"), que en este caso es la página "destino.htm" (por supuesto, si no hemos creado ninguna página llamada "destino.htm", nos dará un error cuando ejecutemos este link).

Pero en el mundo de las páginas dinámicas, nosotros podemos aprovechar la posibilidad de "decirle algo" al servidor junto con el link, para que, además de pedirle ver una página en particular, nos deje ponerle valor a variables que luego estarán disponibles para que las use en esa misma página el intérprete de PHP, cuando le sea "pasada" la página para que la procese antes de devolverla a nuestro navegador.

La forma es la siguiente: supongamos que queremos definir una variable que se llame "nombre" y darle el valor "Pepe". Supongamos también que el link apunta a la página "recibe.php". El link que nos lleve a esa página debería ser así:

Este es el link

Notemos algo, luego de especificar la hiper-referencia (la página "recibe.php") se ha agregado un signo de pregunta. Este signo indica que a continuación le vamos a enviar variables al servidor web.

La forma de pasarle o "definirle" una variable es colocando primero el **nombre** de la variable, luego un **signo igual**, y luego **el valor** que queremos que almacene esa variable (recordar que una variable es como una "cajita" donde depositar un dato por unos instantes)

Podemos probar esto con el siguiente ejercicio:

Vamos a crear un archivo llamado "links.htm" y otro "recibe.php".

El archivo **"links.htm"** será el siguiente:

```
<html>
<head>
<title> Paso de variables por PHP </title>
</head>
<body>
        <a href="recibe.php?nombre=Pepe">Este es el link de Pepe</a><br/>
        <a href="recibe.php?nombre=Pedro">Este es el link de Pedro</a><br/>
        <a href="recibe.php?nombre=Juan">Este es el link de Juan</a>
</body>
</html>
```

Según cuál link pulsemos, estaremos definiéndole un valor distinto a la variable **"nombre**".

El archivo **"recibe.php"** simplemente escribirá el valor que contenga la variable "nombre":

```
<?php
print($nombre);
?>
```

PRINT tiene un uso similar al ECHO, que me permite imprimir en pantalla valores dentro de una variable o texto.

Para que estas variables pasen de esta forma, más las que veremos a continuación, debemos verificar que en el archivo php.ini se encuentre la siguiente línea:

```
register_globals = on
```

En el caso de encontrarse en off, hay que modificarlo, guardar el archivo y volver a iniciar nuestro servidor local (XAMPP, MAMP, etc)

De no tener:

register_globals = on

la página "recibe.php", nos mostrará un error de PHP que nos dirá que las variables que queremos mostrar no existen o no están definidas.

Veremos que según el nombre que escribamos, será lo que veremos en la página "recibe.php".

Además, miremos la barra de direcciones de nuestro navegador, y veremos que la variable pasada y su valor son visibles en la "URL" de la página.

Pero alguien se preguntará: ¿cómo hacemos para pasar más de una variable a la vez en un link?

Simplemente, uniendo cada par "nombredevariable=valor" con un ampersand (&).

Ejemplo de cómo pasar varias variables en un solo link

```
<html>
<head>
<title> Paso de variables por PHP </title>
</head>
<body>
<a href="recibe2.php?nombre=Pepe&apellido=Perez">Este es el link de Pepe</a><br/>
<a href="recibe2.php?nombre=Pedro&apellido=Garcia&edad=9">Este es el link de
Pedro</a><br/>
<a href="recibe2.php?nombre=Juan&edad=30">Este es el link de Juan</a>
</body>
</html>
```

El archivo **"recibe2.php"** será así:

```
<?php
print ("Los valores fueron: ");
print ("<br/>");
print ($nombre);
print ("<br/>");
print ($apellido);
print ("<br/>");
print ($edad);
?>
```

Hemos colocado entre cada variable un "break" (
) para que los valores queden uno en cada línea, y no todos a continuación uno del otro.

-También podemos experimentar el paso de variables directamente **desde el navegador**, escribiendo los nombres de las variables y sus valores en la **barra de dirección** del navegador; esto es muy práctico para chequear qué valores están llegando a la siguiente página y lo usaremos a menudo para detectar errores

http://localhost/recibe.php?nombre=Pepe&edad=9

Cuando pulsemos "enter", esto saldrá enviado al servidor y surtirá el mismo efecto que los links que veníamos haciendo.

A esta forma de pasar variables **en un link** se la conoce con el nombre de "**MÉTODO GET**".

VARIABLE $_GET

Volvamos a tomar un concepto visto en la unidad anterior, las variables supe globales, en este caso la variable ***$_GET.***

La variable $_GET es un vector asociativo, (veremos a fondo vectores en la unidad tres de este mismo módulo) digamos por ahora, muy a grandes rasgos que es un tipo de variable, que guarda toda la información enviada de una página a otra bajo el método "get".

Por ejemplo, si nosotros tenemos un formulario con el valor "get" en la propiedad "method", que contenga los campos con nombre "Usuario" e "Email", al enviar ese formulario podemos ver que nos dirige a la página especificada en en la propiedad "action" con algunas variables en su url.

Ejemplo:
Cuando vemos que tras la url de un documento hay un signo de interrogación, lo que sigue es el nombre de una variable con su valor correspondiente (Variable=valor), y a cada variable a partir de la segunda en vez de un signo de interrogación se le antepondrá un ampersand (&).

La variable $_GET guarda cada una de estas variables, a las que se puede acceder usando el nombre de cada una de ellas como clave del array $_GET.

No sólo se usa la variable $_GET para recuperar la información enviada por un formulario con método "get". Uno mismo puede "filtrar" información en una página enviándole información en la URL.

En el siguiente ejemplo tenemos una página *alumnos.htm* donde hay 2 links.

Ambos links están dirigidos a la misma página, *detallealumnos.php*, con la diferencia de que en el primero se envía la variable nombre=juan y en el segundo nombre=pepe.

Al hacer click en alguno de los dos, nos dirigimos a la página *detallealumnos.php*, donde un script PHP verifica que la variable nombre enviada sea igual a "juan" o "pepe", y depende el caso muestra los datos del alumno correspondiente.

alumnos.htm

```
<a href="detallealumnos.php?nombre=juan">Información de Juan</a>
<a href="detallealumnos.php?nombre=pepe">Información de Pepe</a>
```

detallealumnos.php

```
<?php
if($_GET['nombre']=="juan") { //Evaluamos si la variable "nombre" enviada tiene el valor "juan".
?>
<p>Nombre: Juan</p>
<p>Apellido: Pérez</p>
<p>Edad: 24</p>
<?php
} elseif ($_GET['nombre']=="pepe") { //Evaluamos si la variable "nombre" enviada tiene el valor
"pepe".
?>
<p>Nombre: Pepe</p>
<p>Apellido: González</p>
<p>Edad: 30</p>
<?php
}
?>
```

VARIABLE $_POST

$_POST es otro vector asociativo que también guarda información transferida de una página a otra como la variable $_GET, pero en este caso guarda la información transferida bajo el método “post” de un formulario.

Otra diferencia con $_GET es que las variables enviadas bajo “post” no se verán en la url de la página a la que se envían los datos, por eso es más cómodo usar el método “post” en formularios en vez de “get”.

Formularios

El lenguaje PHP nos proporciona una manera sencilla de manejar formularios, permitiéndonos de esta manera procesar la información que el usuario ha introducido.

Al diseñar un formulario debemos indicar la página PHP que procesará el formulario, así como en método por el que se le pasará la información a la página.

Este es otro método por excelencia para pasar valores entre dos páginas, “el formulario”.

Se trata de un tag del lenguaje HTML, por lo que no es preciso que la página donde está el formulario lleve extensión ".php", puede ser un archivo ".html".

Sus elementos principales son los siguientes:

1) El atributo "***action***", que indica "a qué página" le está pasando las variables, siendo la página que nos va a mostrar cuando pulsemos en el botón "Enviar", tal como si fuese un link hacia esa página.

2) El atributo "***method***", que especifica uno de los dos posibles "métodos" o formas de pasar las variables: a la vista de todos en la URL del navegador (method="GET", el mismo que usamos en la sección anterior para los links que pasan variables) o si las vamos a pasar ocultas (method="POST").
Este último método es el más utilizado en formularios.

3) Algún campo que permita al usuario el ingreso de datos (input). Lo fundamental de cada campo será su nombre (atributo "name"), ya que ése será el nombre de la variable que enviará a la página de destino.

4) Un botón para enviar los datos.

Esos son todos los elementos básicos de un formulario.

Vamos a un ejemplo que llamaremos ***"formu.htm"***

```
<html>
<head>
<title> Paso de variables por PHP </title>
</head>
<body>
        <form action="muestra.php" method="POST">
        <input type="text" name="direccion">
        <input type="submit" value="Enviar">
        </form>
</body>
</html>
```

Y ahora crearemos ***"muestra.php"***, que es la página que recibe la/s variable/s con lo que haya escrito el usuario en el formulario, y la/s muestra:

```
<?php
print ("Su direccion es: ");
print ($direccion);
?>
```

Notemos que ha llegado la variable **$direccion** a esta página mediante el formulario. Cada **<input>** nos genera **una variable**, por lo que si quisiéramos permitir que el usuario escriba varios datos, sólo tenemos que agregar **varios inputs**, de cualquier tipo (texto, textarea multilínea, select desplegable, radio buttons, checkboxes, etc.).

Recomendamos repasar algún manual o tutorial de HTML para refrescar este tema. Ya que a partir de esta unidad nos manejaremos con PHP intercalándolo entre las etiquetas y atributos de HTML

Resumen

En esta Unidad…

En la presente unidad trabajamos con estructuras de control en PHP

En la próxima Unidad…

En la próxima unidad trabajaremos con bucles.