FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA

CURSOS INSTITUCIONALES

PROGRAMACIÓN WEB CON PHP

Del 04 al 22 de Julio de 2005

APUNTES GENERALES

CI - 152

Instructor: Ing. Alejandro Velázquez Mena
DELEGACIÓN AZCAPOTZALCO

JULIO DE 2005

Temario

1. Introducción ... 3
   1.1. Conceptos básicos ... 3
   1.2. El primer programa en PHP ... 3
   1.3. Variables ... 4
   1.4. Operadores Aritméticos ... 5
   1.5. Operadores de Comparación ... 6
   1.6. Operadores Lógicos ... 7
2. Introducción al lenguaje PHP ... 8
   2.1. Condicionales ... 8
      2.1.1 Sentencia if ... else ... 8
      2.1.2 Sentencia switch ... case ... 9
   2.2. Bucles ... 10
      2.2.1 Sentencia while ... 10
      2.2.2 Sentencia for ... 11
   2.3. Salida ... 11
   2.4. Cadenas ... 13
   2.5. Funciones ... 14
   2.6. Librerías ... 15
3. Formularios ... 17
   3.1. Envío y recepción de datos ... 17
   3.2. Method GET y POST ... 18
   3.3. Envío de E-Mail ... 19
4. Trabajando con bases de datos ... 20
   4.1. Crear la base de datos ... 20
   4.2. Conexión a la base de datos ... 21
   4.3. Consultas a la base de datos ... 22
   4.4. Inserción de registros ... 24
   4.5. Borrado de registros ... 26

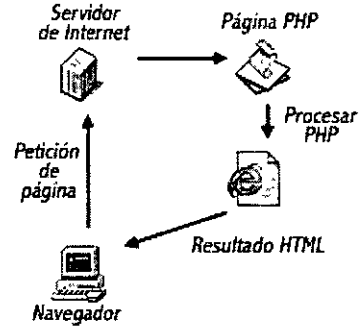
5. Seguridad 27
5.1. Restringir el acceso 27
5.2. Distinción de usuarios 28
6. Sesiones PHP4 30
6.1. ¿Qué son las sesiones? 30
6.2. Inicialización de la sesión 30
6.3. Ejemplo práctico 31
6.4. Error común 32
6.5. Carrito de compra 33
7. Cookies 35
7.1. ¿Qué son las cookies? 35
7.1.1 Funcionamiento 35
7.2. Cómo usar las cookies 36
7.3. Ejemplo de uso de cookies 37

1. Introducción

1.1. Conceptos básicos

El lenguaje PHP es un lenguaje de programación de estilo clásico, con esto quiero decir que es un lenguaje de programación con variables, sentencias condicionales, bucles, funciones.... No es un lenguaje de marcas como podría ser HTML, XML o WML. Está mas cercano a JavaScript o a C, para aquellos que conocen estos lenguajes.

Pero a diferencia de Java o JavaScript que se ejecutan en el navegador, PHP se ejecuta en el servidor, por eso nos permite acceder a los recursos que tenga el servidor como por ejemplo podría ser una base de datos. El programa PHP es ejecutado en el servidor y el resultado enviado al navegador. El resultado es normalmente una página HTML pero igualmente podría ser una pagina WML.



Al ser PHP un lenguaje que se ejecuta en el servidor no es necesario que su navegador lo soporte, es independiente del navegador, pero sin embargo para que sus páginas PHP funcionen, el servidor donde están alojadas debe soportar PHP.

1.2. El primer programa en PHP

La ventaja que tiene PHP sobre otros lenguajes de programación que se **ejecutan en el servidor** (como podrían ser los script CGI Perl), es que nos permite intercalar las sentencias PHP en las paginas HTML, es un concepto algo complicado de entender si no se ha visto nunca como funciona unas paginas PHP o ASP. Vamos a ver un ejemplo sencillo para comprenderlo mejor. En azul está el código HTML y en rojo el código PHP. Seguiremos este criterio durante todo el manual.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
Parte de HTML normal. <BR><BR>
<?php
  echo "Parte de PHP<br>";
  for($i=0;$i<10;$i++)
  {
    echo "Linea ".$i."<br>";
  }
?>
</body>
</html>
```

ejem01.php

El código PHP ejecutado tiene dos partes: la primera imprime "Parte de PHP" y la segunda es un bucle que se ejecuta 10 veces de 0 a 9, por cada vez que se ejecuta se escribe una línea, la variable `$i` contiene el número de línea que se está escribiendo. No importa si no entiende muy bien el programa este ejemplo solo es para ilustrar como se **intercala el código HTML y el código PHP**.

1.3. Variables

Una variable es un contenedor de información, en el que podemos meter números enteros, números decimales, caracteres... el contenido de las variables se puede leer y se puede cambiar durante la ejecución de una página PHP.

En PHP todas las variables comienzan con el símbolo del dólar `$` y no es necesario definir una variable antes de usarla. Tampoco tienen tipos, es decir que una misma variable puede contener un número y luego puede contener caracteres.

```
<html>
<head>
   <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  $a = 1,
  $b = 3.34,
  $c = "Hola Mundo",
  echo $a,"<br>",$b,"<br>",$c,
?>
</body>
</html>
```

ejem02.php

En este ejemplo hemos definido tres variables, $a, $b y $c y con la instrucción echo hemos impreso el valor que contenían, insertando un salto de línea entre ellas. Existen 2 tipos de variables, las variables locales que solo pueden ser usadas dentro de funciones y las variables globales que tienen su ámbito de uso fuera de las funciones, podemos acceder a una variable global desde una función con la instrucción global nombre_variable;

1.4. Operadores Aritméticos

Los operadores de PHP son muy parecidos a los de C y JavaScript, si usted conoce estos lenguajes le resultaran familiares y fáciles de reconocer. Estos son los operadores que se pueden aplicar a las variables y constantes numéricas.

Operador	Nombre	Ejemplo	Descripción
+	Suma	5 + 6	Suma dos números
-	Resta	7 - 9	Resta dos números
*	Multiplicación	6 * 3	Multiplica dos números
/	División	4 / 8	Divide dos números
%	Módulo	7 % 2	Devuelve el resto de dividir ambos números, en este ejemplo el resultado es 1
++	Suma 1	$a++	Suma 1 al contenido de una variable.
--	Resta 1	$a--	Resta 1 al contenido de una variable.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  $a = 8;
  $b = 3,
  echo $a + $b,"<br>";
  echo $a - $b,"<br>";
  echo $a * $b,"<br>";
  echo $a / $b,"<br>",
  $a++,
  echo $a,"<br>";
  $b--;
  echo $b,"<br>",
?>
</body>
</html>
```

ejem03a.php

1.5. Operadores de Comparación

Los operadores de comparación son usados para comparar valores y así poder tomar decisiones.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
==	Igual	$a == $b	$a es igual $b
!=	Distinto	$a != $b	$a es distinto $b
<	Menor que	$a < $b	$a es menor que $b
>	Mayor que	$a > $b	$a es mayor que $b
<=	Menor o igual	$a <= $b	$a es menor o igual que $b
>=	Mayor o igual	$a >= $b	$a es mayor o igual que $b

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  $a = 8;
  $b = 3;
  $c = 3;
  echo $a == $b,'<br>';
  echo $a != $b,"<br>";
  echo $a < $b,"<br>",
  echo $a > $b,"<br>";
  echo $a >= $c,"<br>";
  echo $b <= $c,"<br>";
?>
</body>
</html>
```

ejem03b.php

1.6. Operadores Lógicos

Los operadores lógicos son usados para evaluar varias comparaciones, combinando los posibles valores de estas.

Operador	Nombre	Ejemplo	Devuelve cierto cuando:
&&	Y	(7>2) && (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
and	Y	(7>2) and (2<4)	Devuelve verdadero cuando ambas condiciones son verdaderas.
||	O	(7>2) || (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
or	O	(7>2) or (2<4)	Devuelve verdadero cuando al menos una de las dos es verdadera.
!	No	! (7>2)	Niega el valor de la expresión.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  $a = 8,
  $b = 3,
  $c = 3;
  echo ($a == $b) && ($c > $b),"<br>";
  echo ($a == $b) || ($b == $c),"<br>";
  echo !($b <= $c),"<br>",
?>
</body>
</html>
```

ejem03c.php

2. Introducción al lenguaje PHP

2.1. Condicionales

Las sentencias condicionales nos permiten ejecutar o no unas ciertas instrucciones dependiendo del resultado de evaluar una condición. Las más frecuentes son la instrucción if y la instrucción switch.

2.1.1 Sentencia if ... else

```
<?php
  if (condición)
  {
    Sentencias a ejecutar cuando la
    condicion es cierta
  }
  else
  {
    Sentecias a ejecutar cuando la
    condición es falsa.
  }
?>
```

La sentencia if ejecuta una serie de instrucciones u otras dependiendo de la condición que le pongamos. Probablemente sea la instrucción más importante en cualquier lenguaje de programación.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  $a = 8,
  $b = 3;
  if ($a < $b)
  {
    echo "a es menor que b";
  }
  else
  {
    echo "a no es menor que b";
  }
?>
</body>
</html>
```

ejem04a.php

En este ejemplo la condición no es verdadera por lo que se ejecuta la parte de código correspondiente al else.

2.1.2 Sentencia switch ... case

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  $posicion = "arriba";

  switch($posicion) {
    case "arriba"   // Bloque 1
      echo "La variable contiene",
      echo " el valor arriba";
      break,
    case 'abajo':  // Bloque 2
      echo "La variable contiene",
      echo " el valor abajo";
      break;


    default:  // Bloque 3
      echo "La variable contiene otro valor";
      echo " distinto de arriba y abajo";
  }
?>
</body>
</html>
```

ejem04a2.php

Con la sentencia switch podemos ejecutar unas u otras instrucciones dependiendo del valor de una variable, en el ejemplo anterior, dependiendo del valor de la variable $posicion se ejecuta el bloque 1 cuando el valor es "arriba", el bloque 2 cuando el valor es "abajo" y el bloque 3 si no es ninguno de los valores anteriores.

2.2. Bucles

Los bucles nos permiten **iterar conjuntos de instrucciones**, es decir repetir la ejecución de un conjunto de instrucciones mientras se cumpla una condición.

2.2.1 Sentencia while

```
<?php
  while (condicion)
  {
    intrucciones a ejecutar.
  }
?>
```

Mientras la condición sea cierta se reiterará la ejecución de las instrucciones que están dentro del while.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
Inicio<BR>
<?php
  $i=0;
  while ($i<10)
  {
    echo "El valor de i es ", $i,"<br>",
    $i++;
  }
?>
Final<BR>
</body>
</html>
```

ejem04b.php

En el siguiente ejemplo, el valor de $i al comienzo es 0, durante la ejecución del bucle, se va sumando 1 al valor de $i de manera que cuando $i vale 10 ya no se cumple la condición y se termina la ejecución del bucle.

2.2.2 Sentencia for

```
<?php
  for (inicial ; condición ; ejecutar en iteración)
  {
    intrucciones a ejecutar.
  }
?>
```

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
Inicio<BR>
<?php
  for($i=0 ; $i<10 ; $i++)
  {
    echo "El valor de i es ", $i,"<br>";
  }
?>
Final<BR>
</body>
</html>
```

ejem04b2.php

La instrucción for es la instrucción de bucles más completa. En una sola instrucción nos permite controlar todo el funcionamiento del bucle. El primer parámetro del for, es ejecutado la primera vez y sirve para inicializar la variable del bucle, el segundo parámetro indica la condición que se debe cumplir para que el bucle siga ejecutándose y el tercer parámetro es una instrucción que se ejecuta al final de cada iteración y sirve para modificar el valor de la variable de iteración.

2.3. Salida

Hasta ahora hemos usado la instrucción echo para realizar salida a pantalla, esta instrucción es bastante limitada ya que no nos permite formatear la salida. En esta página veremos la instrucción printf que nos da mucha más potencia.

Sentencia printf

```
<?php
  printf(cadena formato, variable1, variable2...);
?>
```

La cadena de formateo indica cómo se han de representar las valores que posteriormente le indicaremos. La principal ventaja es que además de poder formatear los valores de salida, nos permite intercalar texto entre ellos.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  printf("El numero dos con diferentes formatos: %d %f %.2f",2,2,2);
?>
</body>
</html>
```

ejem04c.php

La cadena de formato puede incluir una seria de caracteres especiales que indican como formatear las variables que se incluyen en la instrucción.

Elemento	Tipo de variable
%s	Cadena de carácteres.
%d	Número sin decimales.
%f	Número con decimales.
%c	Carácter ASCII.
Aunque existen otros tipos, estos son los más importantes.	

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  $var="texto",
  $num=3;
  printf("Puede fácimente intercalar <b>%s</b> con números <b>%d</b> <br>",$var,$num).

  printf("<TABLE BORDER=1 CELLPADDING=20>"),
  for ($i=0;$i<10;$i++)
  {
    printf("<tr><td>%10.d</td></tr>",$i),
  }
  printf("</table>"),
?>
</body>
</html>
```

ejem04c2.php

2.4. Cadenas

Dado el uso del lenguaje PHP el tratamiento de cadenas es muy importante, existen bastantes funciones para el manejo de cadenas, a continuación explicaremos las más usadas.

1. strlen(cadena). Nos devuelve el número de carácteres de una cadena.
2. split(separador,cadena). Divide una cadena en varias usando un carácter separador.
3. sprintf(cadena de formato, var1, var2...). Formatea una cadena de texto al igual que printf pero el resultado es devuelto como una cadena.
4. substr(cadena, inicio, longitud). Devuelve una subcadena de otra, empezando por inicio y de longitud longitud.
5. chop(cadena). Elimina los saltos de línea y los espacios finales de una cadena.
6. strpos(cadena1, cadena2). Busca la cadena2 dentro de cadena1 indicándonos la posición en la que se encuentra.
7. str_replace(cadena1, cadena2, texto). Reemplaza la cadena1 por la cadena2 en el texto.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
  echo strlen("12345"),"<br>",

  $palabras=split(" ","Esto es una prueba"),
  for($i=0,$palabras[$i];$i++)
    echo $palabras[$i],"<br>";

  $resultado=sprintf("8x5 = %d <br>",8*5);
  echo $resultado,"<br>";

  echo substr("Devuelve una subcadena de otra",9,3),"<br><br>";

  if (chop("Cadena \n\n ") == "Cadena")
    echo "Iguales<br><br>";

  echo strpos("Busca la palabra dentro de la frase", "palabra"),"<br><br>";

  echo str_replace("verde","rojo","Un pez de color verde, como verde es la huerba "),"<br>";

?>
</body>
</html>
```

ejem04d.php

2.5. Funciones

El uso de funciones nos da la capacidad de agrupar varias instrucciones bajo un solo nombre y poder llamarlas a estas varias veces desde diferentes sitios, ahorrándonos la necesidad de escribirlas de nuevo.

```
<?php
  function Nombre(parametro1, parametro2...)
  {
    instrucción1,
    instrucción2,
    instrucción3,
    instruccion4,

    return valor_de_retorno,
  }
?>
```

Opcionalmente podemos pasarle parámetros a las funciones que se trataran como variable locales y así mismo podemos devolver un resultado con la instrucción return valor; Esto produce la terminación de la función retornando un valor.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php

  function media_aritmetica($a, $b)
  {
    $media=($a+$b)/2,
    return $media;
  }

  echo media_aritmetica(4,6),"<br>",
  echo media_aritmetica(3242,524543),"<br>';

?>
</body>
</html>
```

ejem05a.php

2.6. Librerías

El uso de librerías es tremendamente útil, nos permiten agrupar varias funciones y variables en un mismo fichero, de manera que luego podemos incluir esta librería en distintas páginas y disponer de esas funciones fácilmente.

```
<?php
  function CabeceraPagina()
  {
?>
  <FONT SIZE="+1">Esta cabecera estará en todas sus páginas.</FONT><BR>
  <hr>
<?
  }
    function PiePagina()
  {
?>
  <hr>
  <FONT SIZE="-1">Este es el pie de página.</FONT><BR>
  Autor. Joaquin Gracia
<?
  }
?>
```

libreria01.php

Ahora vamos a crear 2 páginas que usan la librería definida anteriormente para conseguir que las dos paginas tengan la misma cabecera y pie de pagina. La instrucción para incluir una librería en nuestra página es include("nombre de librería").

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php include("libreria01.php") ?>
<?php CabeceraPagina(); ?>

Página 1
<BR><BR><BR><BR><BR>

Contenido blalbl blalb alb<BR><BR>
más cosas ..<BR><BR>

fin<BR><BR>

<?php PiePagina(); ?>
</body>
</html>
```

ejem05b.php

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php include("libreria01.php") ?>
<?php CabeceraPagina(), ?>

Esta es otra página<BR><BR>
completamente distinta<BR><BR>
pero comparte el pie y la cabecera con la otra.<BR><BR>

<?php PiePagina(), ?>
</body>
</html>
```

ejem05b2.php

3. Formularios

3.1. Envío y recepción de datos

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.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
Introduzca su nombre.
<FORM ACTION="procesa.php" METHOD="GET">
<INPUT TYPE="text" NAME="nombre"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
```

ejem06a.php

Al pulsar el botón Enviar el contenido de cuadro de texto es enviado a la página que indicamos en el atributo ACTION de la etiqueta FORM. En versiones anteriores a 4.2.0 PHP creaba una variable por cada elemento del FORM, esta variable creada tenía el mismo nombre que el cuadro de texto de la página anterior y el valor que habíamos introducido. Pero por razones de seguridad a partir de entonces para acceder a las variables del formulario hay que usar el array de parámetros $_POST[] o $_GET[] dependiendo del método usado para enviar los parámetros. En este ejemplo se ha creado una entrada en el array $_GET[] con el índice 'nombre' y con el valor que haya introducido el navegante.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
El nombre que ha introducido es: <?php echo $_GET['nombre'] ?>
<br>
</body>
</html>
```

procesa.php

3.2. Method GET y POST

En la página anterior hemos comentado que los datos de un formulario se envía mediante el método indicado en el atributo METHOD de la etiqueta FORM, los dos métodos posibles son GET y POST. La diferencia entre estos dos métodos radica en la forma de enviar los datos a la página, mientras que el método GET envía los datos usando la URL, el método POST los envía por la entrada estándar STDIO.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>

<FORM ACTION=" procesa2.php" METHOD="GET">
Introduzca su nombre.<INPUT TYPE="text" NAME="nombre"><BR>
Introduzca sus apellidos:<INPUT TYPE="text" NAME="apellidos"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
```

ejem06b.php

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
<FORM ACTION="procesa2.php" METHOD="POST">
Introduzca su nombre.<INPUT TYPE="text" NAME="nombre"><BR>
Introduzca sus apellidos.<INPUT TYPE="text" NAME="apellidos"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
```

ejem06b2.php

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de procesado de formularios</H1>
El nombre que ha introducido por GET es: <?php echo $_GET['nombre']," ",$_GET['apellidos'] ?><br>
El nombre que ha introducido por POST es: <?php echo $_POST['nombre']," ",$_POST['apellidos'] ?>
<br>
</body>
</html>
```

procesa2.php

El resultado final es el mismo, solo que con el método GET podemos ver los parámetros pasados ya que están codificados en la URL.

3.3. Envío de E-Mail

4. Trabajando con bases de datos

4.1. Crear la base de datos

Para la realización de este curso sobre PHP con acceso a base de datos hemos elegido la base de datos MySQL por ser gratuita y por ser también la mas empleada en entornos UNIX, para lo cual el servidor donde tenemos alojadas las páginas nos tiene que proporcionar herramientas para crearla o acceso al Telnet para que la creemos por nosotros mismos. El comando para crear una base de datos MySQL es el siguiente:

mysqladmin -u root create base_datos

Con este comando conseguimos crear la una base de datos en el servidor de bases de datos de nuestro servidor. Una vez conseguido esto debemos crear las tablas en la base de datos, la descripción de las tablas contienen la estructura de la información que almacenaremos en ellas. Para lo cual usaremos en lenguaje de consultas SQL común para todas las bases de datos relacionales.

En este ejemplo creamos una tabla llamada prueba con 3 campos: un campo identificador, que nos servirá para identificar unívocamente una fila con el valor de dicho campo, otro campo con el nombre de una persona y por último un campo con el apellido de la persona. Para crear la tabla puede usar la herramienta de administración de MySQL de su servidor web o puede escribir un fichero de texto con el contenido de la sentencia SQL equivalente y luego decirle al motor de base de datos que la ejecute con la siguiente instrucción:

mysql -u root base_datos < prueba.sql

```
CREATE TABLE prueba (

ID_Prueba int(11) DEFAULT '0' NOT NULL auto_increment,
Nombre varchar(100),
Apellidos varchar(100),
PRIMARY KEY (ID_Prueba),
UNIQUE ID_Prueba (ID_Prueba)

),
```

prueba.sql

4.2. Conexión a la base de datos

Una vez que tenemos creada la base de datos en nuestro servidor, el siguiente paso es conectarnos a la misma desde una página PHP. Para ello PHP nos proporciona una serie de instrucciones para acceder a bases de datos MySQL.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<?php
function Conectarse()
{
  if (!($link=mysql_connect("localhost","usuario","Password")))
  {
    echo "Error conectando a la base de datos.";
    exit();
  }
  if (!mysql_select_db("base_datos",$link))
  {
    echo "Error seleccionando la base de datos ";
    exit();
  }
  return $link;
}

$link=Conectarse();
echo "Conexión con la base de datos conseguida.<br>";

mysql_close($link); //cierra la conexion
?>
</body>
</html>
```

ejem07a.php

Al ejecutar la instrucción mysql_connect creamos un vínculo entre la base de datos y la pagina PHP, este vínculo será usado posteriormente en las consultas que hagamos a la base de datos. Finalmente, una vez que hemos terminado de usar el vínculo con la base de datos, lo liberaremos con la instrucción mysql_close para que la conexión no quede ocupada.

4.3. Consultas a la base de datos

Una vez que nos hemos conectado con el servidor de bases de datos, ya podemos realizar consultas a las tablas de la base de datos. Para facilitar la programación hemos separado la función de conexión en una librería a parte, de tal manera que la incluiremos en todas las páginas que accedan a la base de datos.

```
<?php
function Conectarse()
{
  if (!($link=mysql_connect("localhost","root","")))
  {
    echo "Error conectando a la base de datos.",
    exit();
  }
  if (!mysql_select_db("base_datos",$link))
  {
    echo "Error seleccionando la base de datos ";
    exit();
  }
  return $link;
}
?>
```

conexion.php

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<?php
  include("conexion.php");
  $link=Conectarse();
  $result=mysql_query("select * from prueba",$link);
?>
  <TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
    <TR><TD> Nombre</TD><TD> Apellidos </TD></TR>
<?php

  while($row = mysql_fetch_array($result)) {
    printf("<tr><td> %s</td><td> %s </td></tr>", $row["Nombre"],$row["Apellidos"]);
  }
  mysql_free_result($result);
  mysql_close($link);
?>
</table>
</body>
</html>
```

ejem07b.php

En este ejemplo hemos utilizado 3 instrucciones nuevas: mysql_query, mysql_fetch_array y mysql_free_result. Con la instrucción mysql_query hemos hecho una consulta a la base de datos en el lenguaje de consultas SQL, con la instrucción mysql_fetch_array extraemos los datos de la consulta a un array y con mysql_free_result liberamos la memoria usada en la consulta.

4.4. Inserción de registros

Hasta ahora nos hemos conectado a una base de datos y hemos hecho consultas a la misma, ahora presentaremos como introducir nuevo registros en la base de datos. Para ello usaremos un formulario y en el ACTION del FORM <FORM ACTION="programaPHP"> indicaremos que debe ser procesado una pagina PHP, esta página lo que hará será introducir los datos del formulario en la base de datos.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>
<FORM ACTION="procesar php">
<TABLE>
<TR>
  <TD>Nombre:</TD>
  <TD><INPUT TYPE="text" NAME="nombre" SIZE="20" MAXLENGTH="30"></TD>
</TR>
<TR>
  <TD>Apellidos:</TD>
  <TD><INPUT TYPE="text" NAME="apellidos" SIZE="20" MAXLENGTH="30"></TD>
</TR>
</TABLE>
<INPUT TYPE="submit" NAME="accion" VALUE="Grabar">
</FORM>
<hr>
<?php
  include('conexion.php'),
  $link=Conectarse(),
  $result=mysql_query("select * from prueba",$link);
?>
  <TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
    <TR><TD> <B>Nombre</B></TD> <TD> ,<B>Apellidos</B> </TD></TR>
<?php

  while($row = mysql_fetch_array($result)) {
    printf("<tr><td> %s</td> <td> %s </td></tr>", $row["Nombre"], $row["Apellidos"]);
  }
  mysql_free_result($result),
  mysql_close($link);
?>
</table>
</body>
</html>
```

ejem07c.php

```
<?php
  include("conexion.php"),
  $link=Conectarse(),
  $nombre=$_GET['nombre'],
  $apellidos=$_GET['apellidos'];
  mysql_query("insert into prueba (Nombre,Apellidos) values ('$nombre','$apellidos')",$link);

  header("Location. ejem07c.php");
?>
```

procesar.php

La primera página PHP ejem07c.php es un formulario que nos permite introducir nombre y apellido para añadirlo a la base de datos, seguido de una consulta que nos muestra el contenido de la tabla prueba. El formulario llama a la pagina procesar.php que añadirá los datos a la tabla.

La segunda página procesar.php se conecta a la base de datos y añade un nuevo registro con la instrucción insert del lenguaje de base de datos SQL. Una vez el registro se ha añadido se vuelve a cargar la página ejem07c.php

4.5. Borrado de registros

Y finalmente, para cerrar el ciclo, nos queda el borrado de registros. El borrado de registros el uno de los procesos más sencillos. Para indicar que elemento vamos a borrar hemos usado un enlace a la página borra.php pasándole el ID_Prueba de cada registro, de esta manera la página borra.php sabe que elemento de la tabla ha de borrar.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de bases de datos con PHP y MySQL</H1>

<?php
  include("conexion.php");
  $link=Conectarse(),
  $result=mysql_query("select * from prueba",$link);
?>
  <TABLE BORDER=1 CELLSPACING=1 CELLPADDING=1>
    <TR><TD> <B>Nombre</B></TD> <TD> <B>Apellidos</B> </TD>
<TD> <B>Borrar</B> ,</TD></TR>
<?php

  while($row = mysql_fetch_array($result)) {
    printf("<tr><td> ,%s</td><td> %s </td><td><a
href=\"borra php?id=%d\">Borra</a></td></tr>", $row["Nombre"],$row["Apellidos"],$row["ID_Prueba"]),
  }
  mysql_free_result($result);
  mysql_close($link),
?>
</table>
</body>
</html>
```

ejem07d.php

```
<?php
  include("conexion php");
  $link=Conectarse();
  $id=$_GET['id'];
  mysql_query("delete from prueba where ID_Prueba = $id",$link),

  header("Location: ejem07d.php' ),
?>
```

borra.php

La página borra.php se conecta a la base de datos y borra el registro indicado en la variable $id que ha sido pasado desde la página ejem07d.php. Una vez el registro se ha borrado se vuelve a cargar la página ejem07d.php

5. Seguridad

5.1. Restringir el acceso

En esta sección vamos a explicar cómo podemos restringir el acceso a según qué páginas, para que solo las personas autorizadas puedan acceder a ciertas partes del nuestro sitio web.

Atención: El acceso restringido a páginas usando las variables globales $_SERVER['PHP_AUTH_USER'], $_SERVER['PHP_AUTH_PW'] y $_SERVER['AUTH_TYPE'] solo funciona si PHP ha sido instalado como un módulo de Apache, si ha sido instalado como un CGI los ejemplos de ésta sección no funcionarán.

Para conseguir la autentificación en las páginas usaremos el sistema de autentificación del protocolo HTTP, este sistema se basa en las variables globales $_SERVER['PHP_AUTH_USER'], y $_SERVER['PHP_AUTH_PW'].

1. $_SERVER['PHP_AUTH_USER'. Nombre de usuario introducido.
2. $_SERVER['PHP_AUTH_PW']. Contraseña introducida.

Para que el navegador nos muestre la ventana de petición de nombre de usuario y contraseña basta con enviar la siguiente cabecera:

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="Acceso restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Authorization Required.';
    exit;
  }
  else {
    echo "Ha introducido el nombre de usuario: {$_SERVER['PHP_AUTH_USER']}<br>";
    echo "Ha introducido la contraseña: {$_SERVER['PHP_AUTH_PW']}<br>";
  }
?>
```

ejem08a.php

Esto provoca que se muestre la ventana de nombre de usuario y contraseña y los datos introducidos se asignen a las variables $_SERVER['PHP_AUTH_USER'], y $_SERVER['PHP_AUTH_PW']. A partir de aquí realizaremos las comprobaciones necesarias para asegurarnos que los datos introducidos son los correctos.

En el siguiente ejemplo pediremos autorización y comprobaremos si el nombre de usuario es Juan y la contraseña 123, si es así tendremos acceso al resto de la página.

```
<?php
  if ((S_SERVER['PHP_AUTH_USER']!="Juan") || (S_SERVER['PHP_AUTH_PW']!="123")) {
    header('WWW-Authenticate: Basic realm="Acceso restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Authorization Required.';
    exit;
  }
?>

<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
Ha conseguido el acceso a la <B>zona restringida</B>.
</body>
</html>
```

ejem08a2.php

5.2. Distinción de usuarios

En la anterior página todo el mundo que tenía acceso a la parte restringida entraba con el mismo nombre de usuario y contraseña, esto evidentemente no es una buena solución, es mejor que cada persona tenga un nombre de usuario y contraseña, ya que de esta forma podemos inhabilitar a un usuario sin ver comprometida la seguridad de nuestro sitio.

En esta página veremos la forma de realizar esto, teniendo un fichero separado con los nombres de usuario y las contraseñas válidas. Dicho fichero podría tener el siguiente formato: nombre_de_usuario | contraseña. Por ejemplo:

```
Joe|1235
Pedro|qwer
Noe|Gty45e
kermit|rwe4v
```

passwords.txt

En este ejemplo se pide la autorización al comienzo de la página si no se ha establecido con anterioridad y se comprueba con el fichero de contraseñas que hemos llamado passwords.txt, si el nombre de usuario y contraseña coincide con alguna entrada del fichero se nos permite ver el resto de la página.

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="Acceso restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Authorization Required.';
    exit;
  }

  $fich = file("passwords.txt");
  $i=0; $validado=false;
  while ($fich[$i] && !$validado) {
    $campo = explode("|",$fich[$i]);
    if ($_SERVER['PHP_AUTH_USER']==$campo[0]) && ($_SERVER['PHP_AUTH_PW']==chop($campo[1]))
$validado=true;
    $i++;
  }

  if (!$validado) {
    header('WWW-Authenticate: Basic realm="Acceso restringido"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Authorization Required.',
    exit;
  }
?>
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
Ha conseguido el acceso a la <B>zona restringida</B> con el usuario
<?php echo $_SERVER['PHP_AUTH_USER']?>.
</body>
</html>
```

ejem08b.php

6. Sesiones PHP4

6.1. ¿Qué son las sesiones?

Si existe una consulta repetida en las listas de PHP, es la relativa al uso de las sesiones. El uso de sesiones es un método ampliamente extendido en cualquier aplicación de cierta entidad. Básicamente una sesión es la secuencia de páginas que un usuario visita en un sitio web. Desde que entra en nuestro sitio, hasta que lo abandona.

El término sesión en PHP, session en inglés, se aplica a esta secuencia de navegación, para ello crearemos un identificador único que asignamos a cada una de estas sesiones de navegación. A este identificador de sesión se le denomina, comúnmente, como la sesión.

El proceso en cualquier lenguaje de programación podría ser algo así:

- ¿Existe una sesión?
- Si existe, la retomamos.
- Si no existe, creamos una nueva.
- Generar un identificador único.

Y para que no perdamos el hilo de la navegación del usuario deberemos asociar esta sesión a todas las URLs y acciones de formulario. Podemos también crear un cookie que incluya el identificador de sesión, pero es conveniente recordar que la disponibilidad o no de las cookies depende del usuario, y no es conveniente fiarse de lo que un usuario pueda o no tener habilitado.

Lo contado hasta ahora es teoría pura y es aplicable a cualquier lenguaje de programación C, Perl, etc. Los que programamos en PHP4 tenemos la suerte de que toda la gestión de sesiones la hace el mismo PHP.

6.2. Inicialización de la sesión

Para utilizar sesiones en PHP lo primero es inicializarlas. Podemos hacerlo explícitamente, mediante la función session_start(), o al registrar una variable en una sesión mediante session_register('miVariable'). En ambos casos se crea una nueva sesión, si no existe, o se retoma la sesión actual. Veamos un sencillo ejemplo:

```
<?php
session_start();
echo "He inicializado la sesión";
?>
```

ejem09a.php

Esta es la forma más básica, si el usuario tiene los cookies activados, PHP habrá insertado de forma automática la sesión y ésta será pasada de una página a otra sin hacer nada más. Desde un punto de vista práctico la sesión es operativa, pero no vemos nada. Podemos obtener la sesión en cualquier momento mediante la función session_id(). Inserta en las sucesivas páginas la siguiente línea para ver si la sesión está disponible:

```
<?php

session_start(),
echo 'La sesión actual es. ' session_id(),
?>
```

ejem09b.php

En este caso session_start() comprueba en los cookies que existe una sesión y continua con ella, session_id() devuelve el identificador actual.

6.3. Ejemplo práctico

Veamos otro ejemplo que, tal vez, te lo aclare un poco más:

```
<?php
session_register('contador'),
echo '<a href="'.S_SERVER['PHP_SELF'].'?' SID.'">Contador vale ' ++S_SESSION['contador']. '</a>';
?>
```

ejem09c.php

Como dije anteriormente la sesión se crea o recoge mediante session_start(), o también cuando se registra una variable de sesión mediante session_register().

Si no has utilizado nunca las sesiones, el concepto de variable de sesión, puede resultar un poco abstracto. Básicamente es una variable, como cualquiera de las que gestiona PHP4, pero que reside en un espacio específico en el servidor, junto con el identificador de sesión, y que pertenece únicamente a un usuario.

En nuestro ejemplo anterior, registramos la variable $contador en la primera línea del script. En la segunda línea, entre otras cosas, cada vez que recarguemos la página o hagamos click sobre el enlace, el valor de $contador se incrementará en 1.

En esta línea hacemos uso de la variable reservada $_SERVER['PHP_SELF'], que hace referencia al propio script en ejecución y una constante propia de PHP4, SID, que contiene el nombre de la sesión y el identificador de la misma.

Podemos averiguar también el nombre de la sesión, o modificarlo, mediante la función session_name(). Veamos una prueba práctica:

```
<?php


session_name('misesion'),
session_register('contador');
echo '<a href="'.$_SERVER['PHP_SELF'].'?'.SID.'">

Contador vale. '.++$_SESSION['contador'].'</a><br>',
echo 'Ahora el nombre es '.session_name().' y la sesión '  $_REQUEST['misesion'].'<br>',


?>
```

ejem09d.php

La asignación del nombre de sesión debe realizarse antes que ninguna otra función con sesiones, antes que session_start() o session_register().

6.4. Error común

Uno de los errores más comunes cuando se utilizan sesiones es dejar líneas en blanco antes de la inicialización de PHP o enviar alguna salida a la pantalla. Para probarlo crea una línea en blanco o con cualquier cosa antes de <?php.

Si tienes los cookies activados, te encontrarás un error de este tipo:

Warning: Cannot send session cookie - headers already sent by (output started at /home/session.php:2) in /home/session.php on line 4

PHP está informando de que no puede activar los cookies en el navegador del usuario, porque las cabeceras ya han sido enviadas. Simplemente por la existencia de una línea en blanco. Como medida práctica, no dejes espacios ni antes del inicio del script, ni después de la finalización.

Si después de todo lo comentado aún no entiendes para que sirven las sesiones, veamos un ejemplo práctico. Imagina que quisieras crear un sistema de cesta de la compra...

6.5. Carrito de compra

Si después de todo lo comentado aún no entiendes para que sirven las sesiones, veamos un ejemplo práctico. Imagina que quisieras crear un sistema de cesta de la compra, en su forma básica podría ser algo así:

```
<?php
session_start();
session_register('itemsEnCesta');
$item=$_POST['item'];
$cantidad=$_POST['cantidad'];
$itemsEnCesta=$_SESSION['itemsEnCesta'];
$encontrado=0;
if ($item){
  if (!isset($itemsEnCesta)){
    $itemsEnCesta[$item]=$cantidad;
  }else{
   foreach($itemsEnCesta as $k => $v){
    if ($item==$k){
     $itemsEnCesta[$k]+=$cantidad;
      $encontrado=1;
      }
  }
   if (!$encontrado) $itemsEnCesta[$item]=$cantidad;
 }
}
$_SESSION['itemsEnCesta']=$itemsEnCesta;
?>
<html>
<body>
<tt>
<form action="<?=$PHP_SELF."?".$SID?>" method="post">
Dime el producto <input type="text" name="item" size="20"><br>
Cuantas unidades <input type="text" name="cantidad" size="20"><br>
<input type="submit" value="Añadir a la cesta"><br>
</form>
<?
if (isset($itemsEnCesta)){
  echo'El contenido de la cesta de la compra es.<br>';
  foreach($itemsEnCesta as $k => $v){
    echo 'Artículo. '.$k.' ud. '.$v.'<br>';
  }
}
?>
</tt>
</body>
</html>
```

ejem09e.php

Una breve explicación. En la línea 4 comprobamos si el usuario ha pasado algún artículo, desde el formulario. En la 5 si el array itemsEnCesta no existe, lo creamos con el nuevo producto y la cantidad indicada. Si el array existe recorremos su contenido, entre las líneas 8 y 13, y si encontramos un artículo igual, añadimos la cantidad en la línea 10. Si no lo encontramos, es un nuevo artículo, por lo tanto, añadimos el nuevo producto con la correspondiente cantidad a itemsEnCesta en la línea 14.

Y a continuación imprimimos el formulario y los resultados, si los hubiera, a partir de la línea 18, donde empieza el HTML. No necesitas archivos, ni bases de datos, ni tienes que andar pasando valores de una página a otra. PHP va gestionando estos datos por nosotros, hasta el momento en que decidamos almacenar la información donde más nos interese.

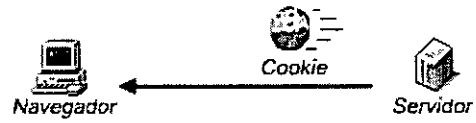
7. Cookies

7.1. ¿Qué son las cookies?

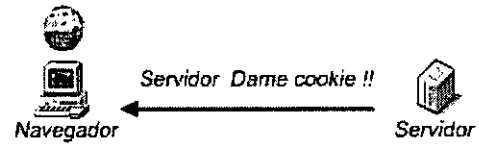
La principal utilidad de las cookies (galletas) es la de solventar el problema de la falta de estado en la navegación a través de las paginas web. Con las cookies, pequeñas porciones de información se quedan registradas en el navegador permitiendo identificar a este a través de diferentes páginas de un mismo sitio e incluso durante visitas entre distintos días. Realmente las cookies no son mas que cadenas de texto que son enviadas desde el servidor al cliente (navegador) y almacenadas en este, luego el navegador envía estas cookies al servidor permitiendo así la identificación del cliente en el servidor.

7.1.1 Funcionamiento

La cookie es enviada al navegador desde el servidor y si este la acepta permanece en él.



Las páginas piden la cookie al navegador...



El navegador las envía, permitiendo la identificación del usuario por parte del servidor.

A continuación vamos a ver como usar las cookies para nuestro beneficio.

7.2. Cómo usar las cookies

El manejo de cookies en PHP se realiza mediante el uso de la función setcookie, esta función esta disponible a partir de la versión 3 de PHP.

```
int setcookie (string Nombre [, string Valor [, int Expire [, string Path [, string Dominio [, int Secure]]]])
```

Setcookie() define una cookie que es enviada junto con el resto de la información de la cabecera(header). Las cookies deben ser enviadas antes de cualquier tag de html, por lo tanto deberemos realizar la llamada a estas funciones antes de cualquier tag <HTML> o <HEAD>. Esta es una restricción de las cookies no de PHP.

Todos los argumentos excepto el nombre son opcionales.

- Nombre. Nombre de la cookie. Si creamos una cookie solamente con el nombre, en el cliente se eliminara la cookie que exista con ese nombre. También podemos reemplazar cualquier argumento con una cadena vacía ("").
- Value. Valor que almacenará la cookie en el cliente.
- Expire. El argumento expire es un argumento entero que indica la hora en que se eliminara la cookie en el formato de hora que devuelven las funciones UNIX time() y mktime(). Normalmente se usa time() + N. segundos de duración, para especificar la duración de una cookie.
- Path. Subdirectorio en donde tiene valor la cookie.
- Dominio. Dominio en donde tiene valor la cookie. Si ponemos como dominio www.domain.com la cookie no se transmite para domain.com, mientras que si ponemos domain.com la cookie se transmite tanto para domain.com como para www.domain.com
- Secure. El argumento secure indica que la cookie solo se transmitirá a través de una conexión segura HTTPS.

```
setcookie("usuario", "Luis", time()+3600,"/","cursophp.com"),
```

En este ejemplo establecemos una cookie de nombre usuario que contiene el valor Luis, que dura 1 hora (3600 segundos) válida para todo el dominio cursophp.com

7.3. Ejemplo de uso de cookies

En este ejemplo vamos a ver como establecer una cookie y cómo se recupera el valor establecido. Para ello pediremos al usuario que introduzca su nombre, que guardaremos en una cookie. Primero pedimos al usuario que introduzca el valor de su nombre, usamos un formulario que procesará la página procesar_cookie.php .

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de cookie</H1>
Introduzca su nombre
<FORM ACTION="procesar_cookie.phtml" METHOD="GET">
<INPUT TYPE="text" NAME="nombre"><BR>
<INPUT TYPE="submit" VALUE="Enviar">
</FORM>
</body>
</html>
```

ejem10a.php

Se establece la cookie ejemusuario con el valor introducido anteriormente, y cuya duración es una hora.

```
<?php
  setcookie("ejemusuario", $_GET['nombre'], time()+3600,"/","");
?>
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de cookie</H1>

Se ha establecido una cookie de nombre <b>ejemusuario</b> con el valor: <b><? print $_GET['nombre'], ?></b> que será
válida durante 1 hora.
</body>
</html>
```

procesar_cookie.php

En este ejemplo vemos lo fácil que es recuperar el valor de la cookie establecida anteriormente.

```
<html>
<head>
  <title>Ejemplo de PHP</title>
</head>
<body>
<H1>Ejemplo de uso de cookie</H1>

Se ha establecido la cookie de nombre <b>ejemusuario</b> vale. <b><? print $_COOKIE['ejemusuario']; ?></b>
</body>
</html>
```

ejem10b.php