

PROGRAMACIÓN EN PHP

Programando con PHP

SINTAXIS BÁSICA: VARIABLES

VARIABLES: ¿QUÉ SON?

- Una variable es un espacio de memoria al que se le da nombre y puede almacenar un determinado valor.
- El valor de las variables puede ser modificado a lo largo del tiempo.
- Para acceder al valor de la variable sólo hay que hacer referencia a su nombre.

VARIABLES

- En PHP las variables se crean en el instante que son utilizadas por primera vez.
- Para inicializarlas se utiliza el operador de asignación (=).
- A partir de ese momento podemos utilizarlas para cualquier cosa.

VARIABLES

- ✖ PHP es un lenguaje débilmente tipado:
 - + una variable podrá almacenar cualquier tipo de información.
- ✖ En PHP todas las variables van precedidas del símbolo \$, seguido de una letra o un (_)

Inválidos	Motivo	Válidos
\$Valor actual	Contiene un espacio	\$Valor_actual
\$#Datos	Contiene carácter no válido	\$NumeroDeDatos
\$2Saldcs	Empieza por número	\$N
\$Prueba,valor	Contiene carácter no válido	\$n

VARIABLES

- ✖ Hay que tener en cuenta que PHP distingue entre mayúsculas y minúsculas.
 - + No es la misma variable \$n que \$N

VARIABLES: EJERCICIOS

✗ Crear una documento PHP en el que se cree una variable llamada “numero”. Asignar a esa variable el valor 15 y mostrarlo por pantalla.

+ Utilizar

- ✗ Color de fondo el color ‘9922AA’
- ✗ Texto tamaño 13 y color blanco

```
<?php
    echo "<b><font color='white' size='13'>";
    $numero = 15;
    echo $numero;
    echo "</font></b>";
?>
```

VARIABLES: EJERCICIOS

- ✖ Crear una página Web en PHP en la que haya una única variable. Esta variable deberá contener los siguientes valores:
 - + 3
 - + Casa
 - + 3,254
 - + Hola mundo
- ✖ Mostrar dichos valores en líneas independientes

```
1 <html>
2     <head>
3         <title> PHP </title>
4     </head>
5     <body>
6         <?php
7             $variable = 3;
8             echo "variable vale: $variable";
9             echo "<br>";
10            $variable = "casa";
11            echo "variable vale: $variable";
12            echo "<br>";
13            $variable = 3.254;
14            echo "variable vale: $variable";
15            echo "<br>";
16            $variable = "hola mundo";
17            echo "variable vale: $variable";
18            echo "<br>";
19        ?>
20        </font>
21    </body>
22 </html>
```

VARIABLES: EJERCICIOS

- Escribir un programa que nos pregunte nuestro nombre y nos salude.
- NOTA: por ahora no podemos pedir al usuario que nos dé un dato: simularemos la petición de datos asignando directamente el valor a una variable.

```
<?php  
    $nombre='Pepe';  
    echo "Hola $nombre";  
?>
```

```
<?php  
    $nombre='Pepe';  
    echo "Hola".$nombre;  
?>
```

VARIABLES: EJERCICIOS

- ✗ Escribir un programa que nos pregunte nuestro nombre y nos salude dándonos la bienvenida a nuestra web.

```
<?php  
    $nombre = 'Pepe';  
    echo "Hola $nombre bienvenido a nuestra website";  
?  
?>
```

```
<?php  
    $nombre = 'Pepe';  
    echo "Hola ".$nombre." bienvenido a nuestra Website";  
?  
?>
```

VARIABLES: EJERCICIOS

- Escribir un programa que nos pida insertar una palabra. El programa deberá mostrar por pantalla la palabra subrayada.

```
<?php  
    $palabra = 'margarita';  
    echo "La palabra insertada es <u>$palabra</u>";  
?>
```

```
<?php  
    $palabra = 'margarita';  
    echo "La palabra insertada es <u>".$palabra."</u>";  
?>
```

Programando con PHP

SINTAXIS BÁSICA: TIPOS DE DATOS

TIPOS DE DATOS

TIPOS DE DATOS

✗ Numéricos

- + Permite almacenar números que van desde los -2 billones hasta 2 billones.
- + Para almacenar en una variable un valor numérico se asigna el valor utilizando el operador de asignación (=).

TIPOS DE DATOS

TIPOS DE DATOS

✗ Cadenas

- + Representan a un conjunto de 0 o más caracteres.
- + Se encierran entre (“ ”) o (‘ ’).

✗ Con “”:

- + Podremos incluir en la cadena nombres de variables (en ese caso se mostrará el valor de la variable).
- + Podremos incluir en la cadena ‘’.

✗ Con ‘’:

- + No se evaluará nada de lo que aparezca en la cadena.

<H2>Trabajando Con Caderas de caracteres </H2>

```
<?php
```

```
    $lenguaje = "PHP";
    $ver = "v5";
    echo "<B>Estamos 'trabajando' con $lenguaje $ver"
    echo "</B><BR><BR>";
    echo 'La variable $lenguaje contiene: ';
    echo "$lenguaje <BR>";
    echo 'La variable $ver contiene: ';
    echo $ver;
```

```
?>
```

Trabajando Con Caderas de caracteres

Estamos 'trabajando' con PHP v5

La variable \$lenguaje contiene: PHP

La variable \$ver contiene: v5

Comillas dobles y simples

```
<?php  
    $cadena=' un tesoro ';  
    echo "la cadena '$cadena' contiene: $cadena"  
    echo "<br><br>";  
    echo "La cadena \"\$cadena\" contiene: $cadena"  
    echo "<br><br>";  
    echo 'La cadena "$cadena" contiene: $cadena'  
    echo '<br><br>';  
?>
```

Comillas dobles y simples

la cadena ' un tesoro ' contiene: un tesoro

La cadena " un tesoro " contiene: un tesoro

La cadena "Scadena" contiene: Scadena

Programando con PHP

SINTAXIS BÁSICA: OPERADORES

OPERADORES ARITMÉTICOS

Operador	Ejemplo	Descripción
+	$$a + b	Suma dos operandos
-	$$a - b	Resta dos operandos
*	$$a * b	Multiplica dos operandos
/	$$a / b	Divide dos operandos
%	$$a \% b	Resto de la división entera

OPERADORES DE ASIGNACIÓN

Operador	Ejemplo	Descripción
=	$\$a = \b	$\$a$ toma el valor de $\$b$
+=	$\$a += \b	Equivale a $\$a = \$a + \$b$
-=	$\$a -= \b	Equivale a $\$a = \$a - \$b$
/=	$\$a /= \b	Equivale a $\$a = \$a / \$b$
*=	$\$a *= \b	Equivale a $\$a = \$a * \$b$
%=	$\$a %= \b	Equivale a $\$a = \$a \% \$b$
.=	$\$a .= \b	Equivale a $\$a = \$a . \$b$

OPERADORES DE INCREMENTO Y DECREMENTO

Operador	Ejemplo	descripción
++	++\$a	Preincremento: incrementa \$a en uno y después devuelve \$a
	\$a++	Postincremento: devuelve \$a y después incrementa en uno \$a
--	-\$a	Predescuento: decremente en uno \$a y después devuelve \$a
	\$a--	Postdescuento: devuelve \$a y después decremente en uno \$a

OPERADORES DE COMPARACIÓN

Operador	Ejemplo	Devuelve TRUE cuando
<code>==</code>	<code>\$a == \$b</code>	Los operandos son iguales
<code>!=</code>	<code>\$a != \$b</code>	Los operandos son distintos.
<code>====</code>	<code>\$a === \$b</code>	Los operandos son idénticos: iguales y del mismo tipo.
<code>!==</code>	<code>\$a !== \$b</code>	Los operandos no son iguales o del mismo tipo.
<code><</code>	<code>\$a < \$b</code>	El operando de la izquierda es menor que el de la derecha
<code>></code>	<code>\$a > \$b</code>	El operando de la izquierda es mayor que el de la derecha
<code><=</code>	<code>\$a <= \$b</code>	El operando de la izquierda es menor o igual que el de la derecha
<code>>=</code>	<code>\$a >= \$b</code>	El operando de la izquierda es mayor o igual que el de la derecha

OPERADORES LÓGICOS

Operador	Ejemplo	Devuelve TRUE cuando
&&	<code>\$a && \$b</code>	\$a y \$b son ambos true
And	<code>\$a and \$b</code>	\$a o \$b son true
	<code>\$a \$b</code>	\$a es false, niega el valor lógico de la variable
Or	<code>\$a or \$b</code>	\$a es true o \$b es true, pero no lo son los dos a la vez
!	<code>!\$a</code>	
xor	<code>\$a xor \$b</code>	

VARIABLES: EJERCICIOS

- ✖ Crear una pagina Web en la que se creen 2 variables una llamada numero1 y otra llamada numero2.
 - + En la variable numero1 meteremos el valor 3
 - + En la variable numero2 meteremos el valor 4
 - + En la página Web se mostrarán los valores de ambas variables y el resultado de sumarlas.

```
1 <html>
2     <head>
3         <title> PHP </title>
4     </head>
5 <body>
6     <?php
7         $numero1 = 3;
8         $numero2 = 4;
9         echo "numero 1 vale: $numero1";
10        echo "<br>";
11        echo "numero 2 vale: $numero2";
12        echo "<br>";
13        $suma = $numero1 + $numero2;
14        echo "la suma de los dos valores es: $suma";
15
16
17    ?>
18 </body>
19 </html>
```

VARIABLES: EJERCICIOS

- Se necesita obtener la nota media de un estudiante a partir de sus tres notas parciales.

```
<?php  
    $nota1 = 7;  
    $nota2 = 5;  
    $nota3 = 9;  
  
    $notaMedia = ($nota1 + $nota2 + $nota3)/3;  
  
    echo "El resultado de la nota media es: $notaMedia";  
?  
?>
```

EJERCICIOS

- ✖ Crear una página Web en PHP que tome dos números y muestre la suma de ellos.
 - + Se deberán mostrar los valores de los dos números y el de la suma
- ✖ Crear una Web en PHP que con dos números cuales quiera muestre una lista de la siguiente forma:
 - Suma -> $25+13 = 38$
 - Resta -> $25-13 = 12$
 - Multiplicación -> $25*13 = 325$
 - División -> $25/13 = 1.9230$

EJERCICIOS

✗ Crear un documento PHP que muestre como salida la tabla de multiplicar del 5.

+ Se deberá:

✗ Hacer dentro de una lista como la siguiente:

- $5 \times 1 = 5$
- $5 \times 2 = 10$
- $5 \times 3 = 15$
- $5 \times 4 = 20$
- $5 \times 5 = 25$
- ...
- $5 \times 10 = 50$

✗ Utilizar una única variable

Programando con PHP

LOS ARRAYS

ARRAYS

- ✖ Conjunto de datos almacenados bajo el mismo nombre.
- ✖ Son posiciones de memoria consecutivas, todas ellas referenciadas por el mismo nombre.
- ✖ Un array en PHP no tiene un tamaño predefinido, con lo que puede tener tantos valores como sea necesario.

ARRAYS

- Podemos acceder a cada uno de los elementos del array haciendo referencia a la posición que ocupan dentro del conjunto de valores.
- Las posiciones de un array siempre empiezan por 0.

Posición	0	1	2	3
Valor	25	36	21	58

ARRAYS

- ❖ Puesto que PHP es un lenguaje débilmente tipado, los arrays pueden estar compuestos por elementos de diferente tipo.

Posición	0	1	2	3	4
Valor	León	Seat	2.500	V6	

ARRAYS

- La declaración de un array posicional se hace como si cada posición fuera una variable independiente.
- Para indicar a qué posición queremos hacer referencia utilizamos los corchetes ([])

```
7 <tr>
8     <td> Posición </td>
9     <td> 0 </td>
10    <td> 1 </td>
11    <td> 2 </td>
12    <td> 3 </td>
13 </tr>
14 <tr>
15     <td> Valor </td>
16 <?php
17     $array[0]=25;
18     $array[1]=36;
19     $array[2]=21;
20     $array[3]=58;
21     echo "<td> $array[0] </td>";
22     echo "<td> $array[1] </td>";
23     echo "<td> $array[2] </td>";
24     echo "<td> $array[3] </td>";
25 ?>
26 </tr>
```

ARRAYS

- Para asignar un valor a la última posición de un array no hace falta indicar nada entre los corchetes.

```
<?php  
    $matriz1[0]="rojo";  
    $matriz1[1]="azul";  
    $matriz1[2]="violeta";  
    $matriz1[3]="";  
    $matriz1[]="rosa";  
    $matriz1[]="amarillo";  
?  
?>
```

EJERCICIOS

- ✖ Crear un array de 4 posiciones y mostrar los valores de la siguiente forma:
 - + En la posición 1 hay un: *número*
 - + En la posición 2 hay un: *número*
 - + En la posición 3 hay un: *número*
 - + En la posición 4 hay un: *número*
- ✖ Crear un array de 3 posiciones, asignarle a cada posición un valor. Después mostrar por pantalla el siguiente mensaje:
 - + *Valor1 + valor2 + valor3 = suma*

EJERCICIOS

- ✖ Crear un array con los nombre de los alumnos de una clase (al menos 5 alumnos) y mostrarlos de la siguiente forma:
 - + Los alumnos de la clase son:
 - ✖ Nombre1
 - ✖ Nombre2
 - ✖ Nombre3
 - ✖ Nombre4

EJERCICIOS

- ✖ Crear un array en el que estén los datos personales de alguien (Nombre, apellidos, edad y DNI) y mostrarlos de la siguiente forma:
 - + Los datos personales de *Nombre* son:
 - ✖ Apellidos: *apellidos*
 - ✖ Edad: *edad*
 - ✖ DNI: *DNI*

EJERCICIOS

EJERCICIOS

- Crear el siguiente array posicional y mostrarlo dentro de una tabla igual a la siguiente:

0	1	2	3	4	5	6
Jorge	Pérez	35	1.77	80	Moreno	Soltero

EJERCICIOS

- ✖ Crear un array que almacene los días de la semana y mostrarlos dentro de una lista.
- ✖ Crear un array que almacene los siguientes valores:
 - + 8
 - + Hola
 - + 25.6
 - + Adios
- ✖ Mostrar el array en orden inverso.

MATRICES

¿QUÉ SON LAS MATRICES?

- ✖ Son arrays bidimensionales, es decir, arrays que se organizan en filas y columnas.
- ✖ Para acceder a cada posición de una matriz tendremos que referenciarla con 2 índices: fila y columna en la que está situado.
- ✖ \$matriz[1][2]=5
- ✖ \$matriz[2][2]=9
- ✖ \$matriz[1][3]=43

2	85	32	10
68	23	5	43
25	12	9	0

EJERCICIOS

EJERCICIOS

- ✖ Crear una matriz de 3x5 que contenga 15 números y mostrar:
 - + Sólo los pertenecientes a la 3 fila.
 - + Sólo los pertenecientes a la 1 columna.
- ✖ Crear una matriz de 4x3 en la que se van a almacenar el nombre, apellido y edad de 4 personas. Mostrarlo después como la siguiente tabla:

Pepe	Lopez	35
Maria	Fernandez	15
Paloma	Ruiz	68
Pedro	Molina	24

EJERCICIOS

EJERCICIOS

- ✗ Crear una matriz de 3 filas y 2 columnas, mostrar después la siguiente tabla.

Numero1	4	8
Numero2	4	6
Numero3	2	9
Suma	10	23

- ✗ Crear una matriz de 5x5 y rellenarla con letras. Después, pedir al usuario qué casilla quiere visualizar y mostrar el siguiente mensaje:
 - + En la casilla *filaxcolumna* está la letra: *letra*

Programando con PHP

LAS CONSTANTES

CONSTANTES

- Una constante es un tipo especial de identificador.
- Admite un valor único que no podrá cambiar durante la ejecución del programa.
- Permite mantener un valor fijo bajo un nombre sin tener el riesgo de que se modifique durante la ejecución de el programa.

CONSTANTES

- ✖ Crear una constante en PHP es tan fácil como usar una de las siguientes funciones:
 - + Define(nombre_constante, valor): crea la constante con el valor.
- ✖ Para hacer referencia a una constante no hay que poner \$ delante de su nombre.

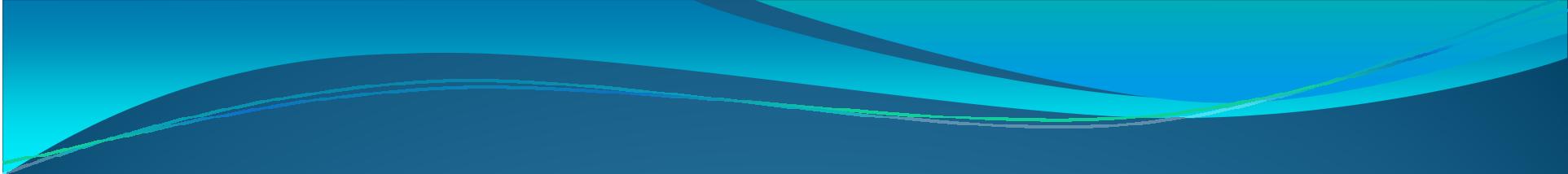
EJERCICIOS

- ✗ Crear un documento PHP que contenga una constante de nombre PRUEBA. Al crearla asignarle un valor numérico cualquiera. Comprobar cómo al intentar cambiar el valor PHP nos devuelve un error.

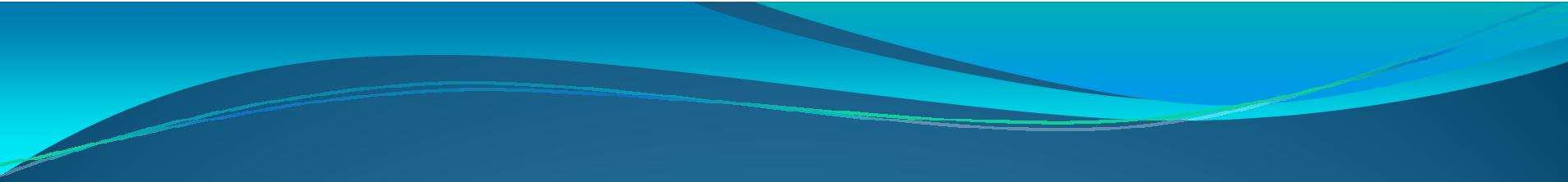
CONSTANTES PREDEFINIDAS

Constante	Significado
PHP_VERSION	Cadena que representa la versión del intérprete de PHP en uso.
PHP_OS	Cadena con el nombre del sistema operativo en el que se está ejecutando el intérprete de PHP
TRUE	Verdadero
FALSE	Falso
E_ERROR	Información sobre errores distintos a los de interpretación del cual no es posible recuperarse.
E_PARSE	Informa que el intérprete encontró una sintaxis inválida en el archivo de comandos. Finaliza la ejecución.
E_NOTICE	Informa que se produjo algo incorrecto que puede provenir o no de un error. A ejecución continúa.
E_WARNING	Denota un error que no impide que continúe la ejecución.
E_ALL	Conjunto con todos los errores que se han producido.

PROGRAMACIÓN EN PHP



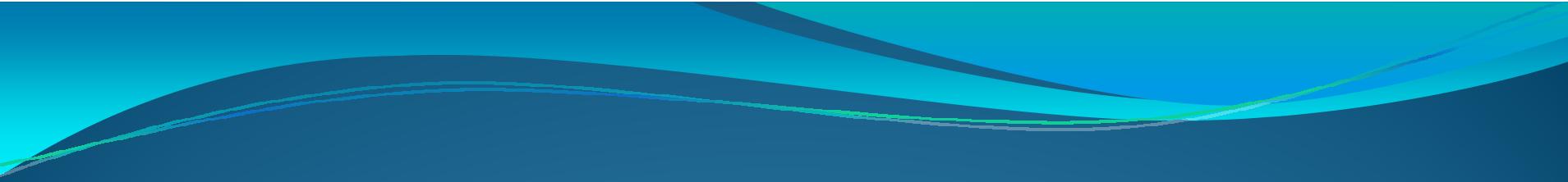
Bloque 0: Introducción



¿Qué es PHP?

¿Qué es PHP?

- PHP es el acrónimo recursivo de Hypertext Preprocessor
- Es un lenguaje de código abierto muy popular
- Inicialmente usado para crear aplicaciones.
- Popularizado para desarrollo de aplicaciones Web
- Se suele utilizar incrustado en HTML



¿Qué podemos hacer con PHP?

¿Qué podemos hacer con PHP?

- PHP es una herramienta de desarrollo Web que permite:
 - Procesar información de formularios.
 - Generar páginas con contenidos dinámicos
 - Enviar y gestionar Cookies

¿Qué podemos hacer con PHP?

- Permite mejorar un sitio Web añadiéndole dinamismo, interacción con el usuario y acceso a Bases de Datos.
 - **Dinamismo:** los sitios Web van a cambiar de forma y de información dependiendo del momento en que se consulten.
 - **Interacción con el usuario:** desde el sitio Web el usuario podrá hacer peticiones y se resolverán dichas peticiones mostrando el resultado en la Web.
 - **Acceso a Bases de Datos:** se almacenará información en Bases de Datos que serán accesibles al usuario desde los sitios Web

Fucionamiento de PHP

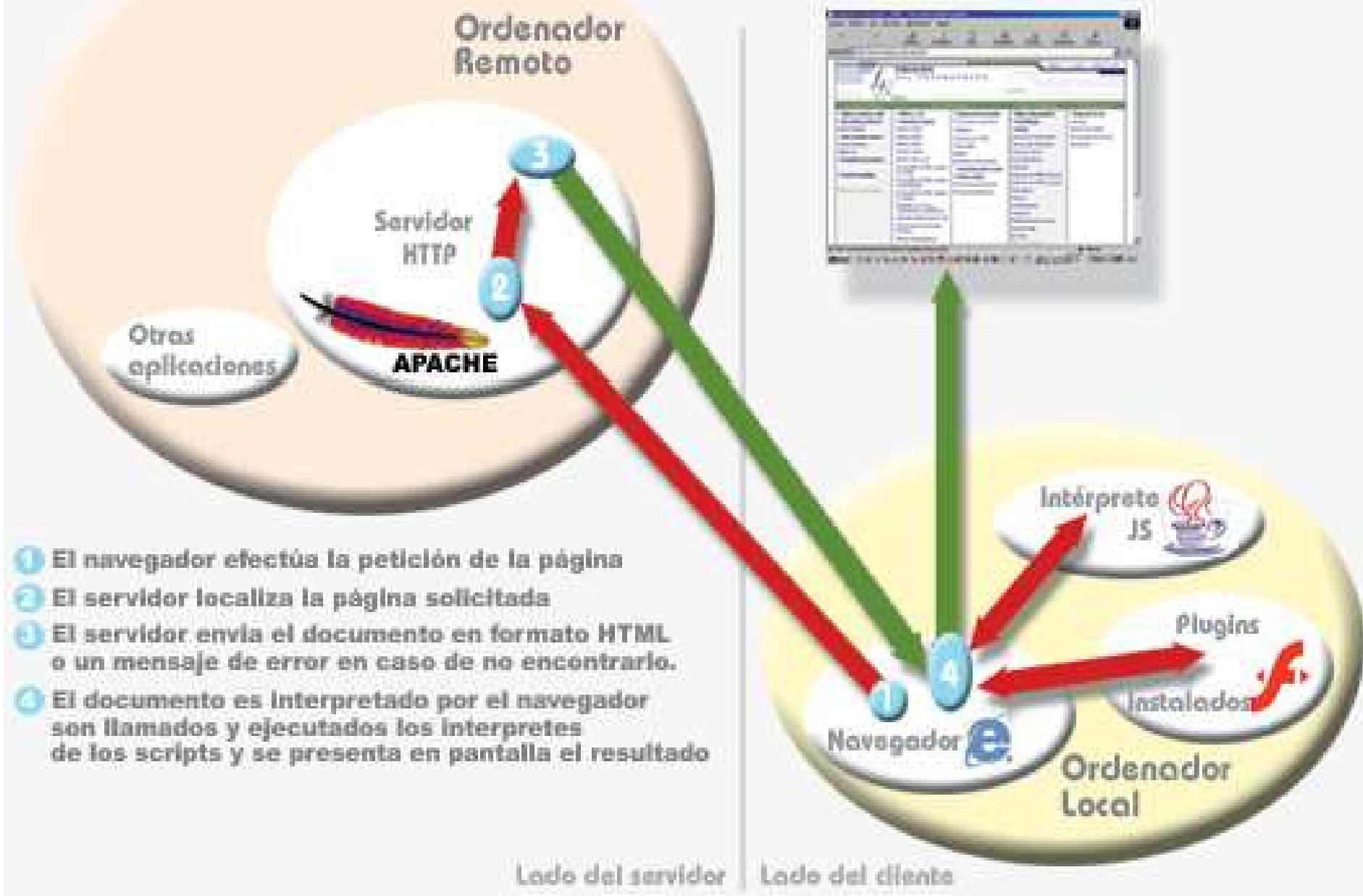
Funcionamiento de PHP

- Es un lenguaje de programación del lado del Servidor.
- Todos los documentos escritos en PHP estarán almacenados en un Servidor Web.
- Los clientes Web (navegador) no entienden las instrucciones escritas en PHP.
- ¿Cómo funcionan entonces los sitios escritos en PHP?

Funcionamiento de PHP

- Cuando un sitio Web está escrito únicamente con HTML el funcionamiento es el siguiente:
 - El cliente (navegador) solicita el documento.
 - El servidor busca en su directorio el documento solicitado.
 - El servidor envía al cliente el documento.
 - El cliente interpreta las instrucciones HTML del documento y muestra por pantalla el resultado.

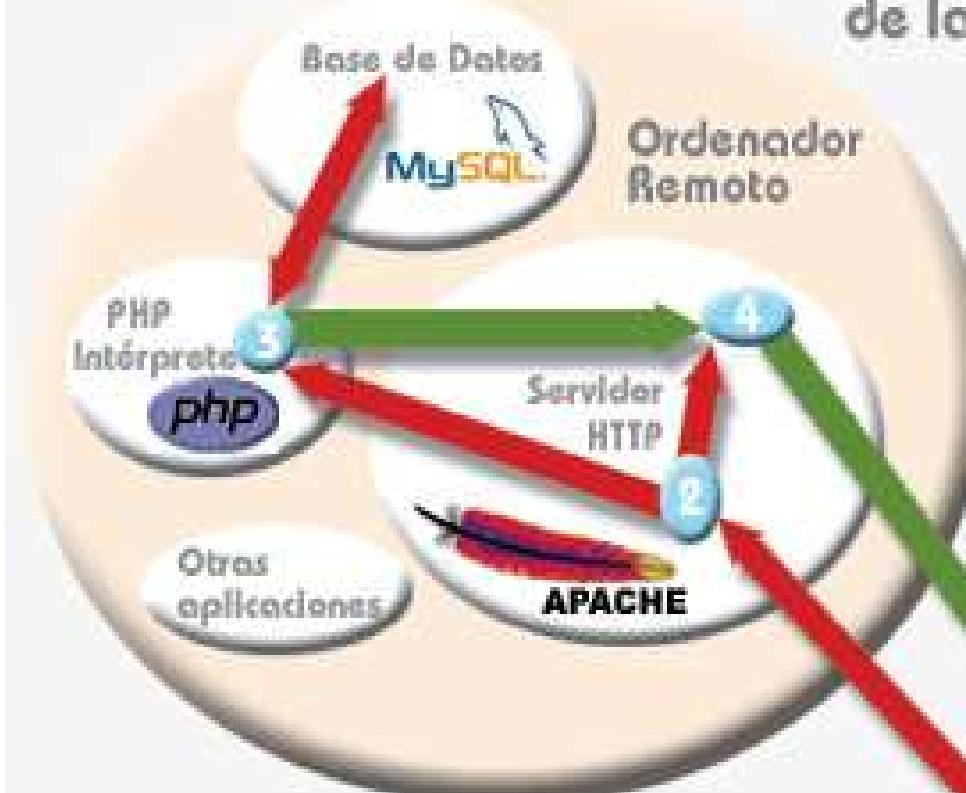
Páginas dinámicas usando únicamente aplicaciones del lado del cliente



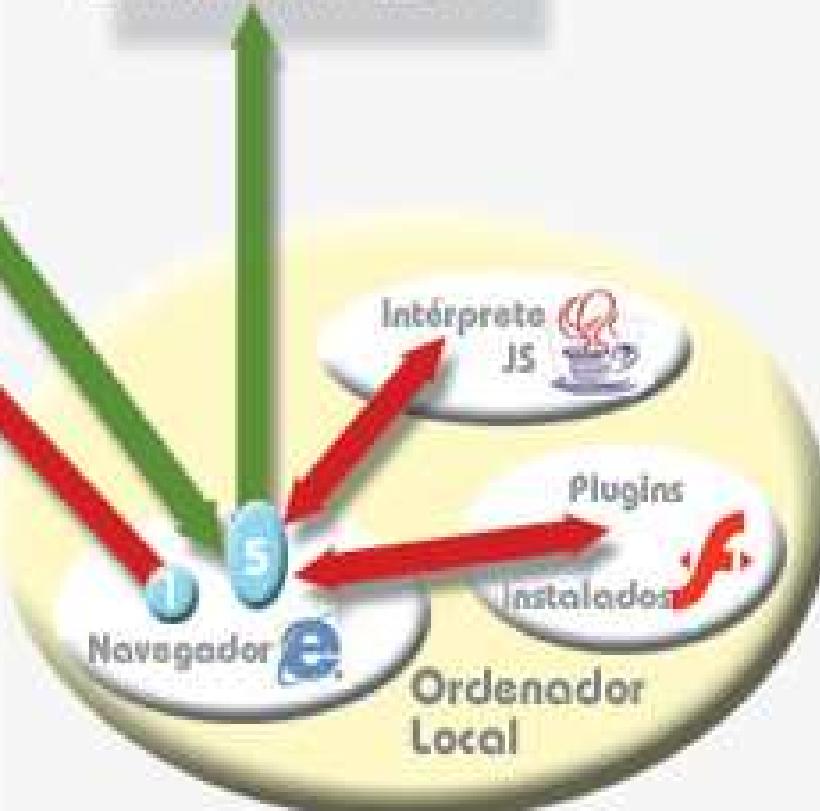
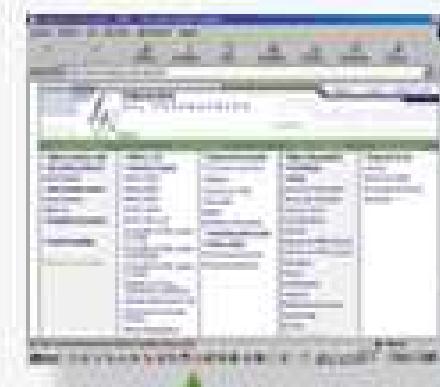
Funcionamiento de PHP

- Cuando un sitio Web está escrito con instrucciones PHP:
 - El cliente (navegador) solicita el documento.
 - El servidor busca en su directorio el documento solicitado.
 - *El servidor traduce las instrucciones PHP a sus equivalentes en HTML*
 - El servidor envía al cliente el documento *traducido*.
 - El cliente interpreta las instrucciones HTML del documento y muestra por pantalla el resultado

Páginas dinámicas usando aplicaciones de lado del servidor y del lado del cliente



1. El navegador efectúa la petición de la página.
2. El servidor llama al interprete del PHP si es necesario.
3. PHP ejecuta los scripts (interactuando con la base de datos si es preciso) y devuelve al servidor el documento generado.
4. El servidor envia el documento resultante en formato HTML.
5. El documento es Interpretado por el navegador se ejecutan los scripts del lado del cliente y se presenta el resultado en pantalla.



Lado del servidor | Lado del cliente

Instalación de un Servidor Web

Instalación de un servidor Web: XAMPP

- Nosotros vamos a utilizar el servidor XAMPP Server.
 - X: funciona sobre cualquier sistema Operativo.
 - A: es un servidor Apache
 - M: utiliza MySQL como SGBD.
 - P: utiliza PHP como lenguaje de programación.
 - P: permite también Perl como lenguaje de programación



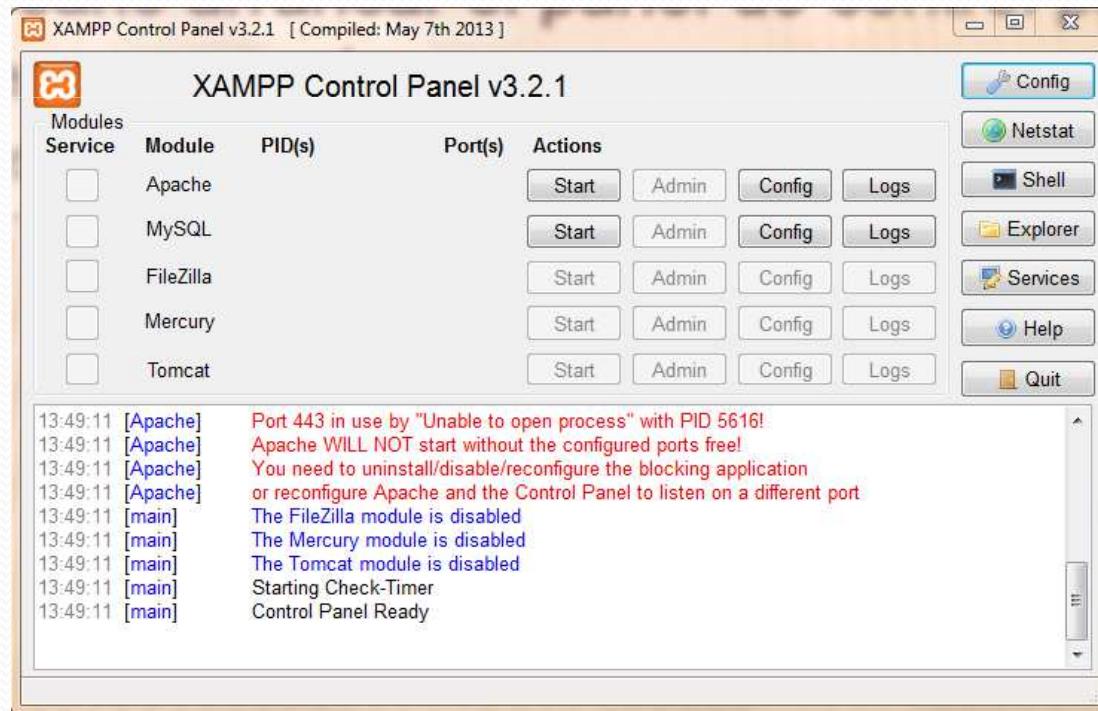
Instalación de un servidor Web: XAMPP

- Además Xampp Server es gratuito.
- Al ser código abierto, se puede descargar de cualquier página dónde lo tengan. Recomendada:

www.apachefriends.org/es/xampp.html

Arrancando el servidor

- Una vez instalado XAMPP Server es necesario arrancar el panel de control para poder arrancar los programas que necesitamos.



Los documentos PHP

- Para que el servidor entienda que el documento solicitado contiene código PHP, la extensión de dicho documento será .php
- El documento deberá estar localizado en el directorio del servidor, normalmente:

C:/xampp/htdocs

Fundamentos del lenguaje PHP

Programando con PHP

Fundamentos de PHP

- La sintaxis de PHP es muy parecida a otros lenguajes de programación como C o JAVA.
- Tiene un vocabulario relativamente pequeño y fácil de comprender.
- Ofrece las características básicas de un lenguaje de programación.
 - Bucles
 - Sentencias de control
 - Etc.

Formato del código PHP

- **Delimitadores:**

- El código PHP va incrustado en HTML por lo que necesita de etiquetas para ser encontrado:
- Las etiquetas son:

```
<?php  
      instrucciones PHP  
?>
```

Comentarios

- Los comentarios son líneas de texto que no queremos que el servidor interprete.
- Existen en PHP varios tipos de comentarios
 - Comentario de una sola línea.
 - //
 - Comentario de varias líneas.
 - /* */
- **NOTA IMPORTANTE:** Todas las líneas de PHP deben terminar con punto y coma (;

La función ECHO

- Muestra por pantalla aquello que le pidamos.

```
<?php  
    echo "texto a mostrar";  
?>
```

```
1 <html>
2   <head>
3     <title> Prueba Apache </title>
4   </head>
5   <body>
6     <center>
7       <h2>
8         
9         <?php
10        // esto es un comentario de una linea
11        echo "El módulo de PHP funciona correctamente";
12        /* esto es un comentario
13        de varias líneas
14        ninguno de los dos se interpretará*/
15      ?>
16     </h2>
17     </center>
18   </body>
19 </html>
```



Fig. 3. Ejecución de prueba1.php

Ejercicios

Ejercicios

- Crear una página en PHP que muestre el saludo: “hola mundo”. La página deberá tener
 - Fondo rojo
 - Texto tamaño 7 y color verde.

```
1  <HTML>
2      <HEAD>
3          <TITLE> Ejercicio 1 PHP </TITLE>
4      </HEAD>
5      <BODY bgcolor='red' text='green'>
6          <font size=7>
7              <?php
8                  echo 'HOLA MI MUNDO'
9              ?>
10             </font>
11         </BODY>
12     </HTML>
```

Acceder a una página

- El navegador no puede interpretar el lenguaje PHP.
- Doble clic solo visualiza la parte de HTML.
- Para acceder a una página escrita con PHP
 - Hay que solicitarla al servidor.
 - Servidor local: localhost

Ejercicios

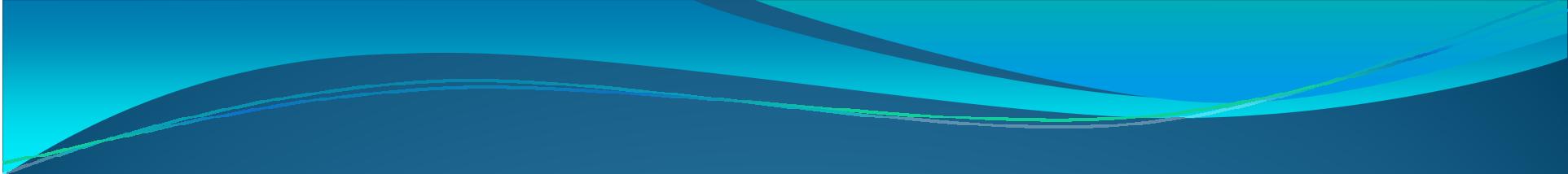
- Crear una página en PHP que muestre el saludo: “hola mundo”. La página deberá tener
 - Fondo rojo
 - Texto tamaño 7 y color verde.
- Crear una página en PHP que muestre dos mensajes “Hola Amigo” y “Bienvenido a mi Web”
 - Los mensajes deberán visualizarse en dos líneas distintas.

```
1 <html>
2     <head>
3         <title> PHP </title>
4     </head>
5     <body bgcolor='#DDDDDD'>
6         <font size='7' color='#33FF99'>
7             ...
8             <?php
9                 echo "hola mundo";
10                echo "<br>";
11                echo "Bienvenido a mi Web";
12            ?>
13        </font>
14    </body>
15 </html>
```

Ejercicios

- Crear una página en PHP que muestre la siguiente lista:
 - **Lunes:**
 - Entrada: 0900
 - Salida: 13.00
 - **Martes**
 - Entrada: 11.00
 - Salida: 15.00
 - **Miércoles:**
 - Entrada: 09.30
 - Salida: 12.30
- Nota: hacer la lista desde una sección PHP

```
<?php  
echo "<ul>";  
echo "<li><b><font color='purple'>Lunes: </font></b></li>";  
echo "<ul>";  
echo "<li><u>Entrada:</u> 09.00</li>";  
echo "<li><u>Salida:</u> 13.00</li>";  
echo "</ul>";  
echo "<li><b><font color='purple'>Martes: </font></b></li>";  
echo "<ul>";  
echo "<li><u>Entrada:</u> 11.00</li>";  
echo "<li><u>Salida:</u> 15.00</li>";  
echo "</ul>";  
echo "<li><b><font color='purple'>Miércoles: </font></b></li>";  
echo "<ul>";  
echo "<li><u>Entrada:</u> 09.30</li>";  
echo "<li><u>Salida:</u> 12.30</li>";  
echo "</ul>";  
echo "</ul>";  
?>
```



Bloque 0: Introducción

Programación en PHP

Estructuras condicionales

Sentencias condicionales

- Hay ocasiones en que la ejecución de un bloque PHP depende de que ocurra o no algo → Condición
- Según esto: si la condición se cumple se hará algo y si no se cumple pasará otra cosa

La sentencia IF

- Con la sentencia IF indicamos que queremos que ocurra si se cumple una condición.
- Si no se cumple, el programa seguirá su curso normal.
- Sintaxis:

```
If (condición)
{
    sentencias;
}
```

La sentencia IF: Ejemplos

- Escribir un sitio Web que compruebe si el contenido de una variable es "a". Si es así, el sitio mostrará un mensaje indicándolo.

```
<?php  
    $letra = 'a';  
    if ($letra=='a')  
    {  
        echo "La variable SI contiene la letra a";  
    }  
?>
```

La variable SI contiene la letra a

```
<?php  
    $letra = 'z';  
    if ($letra=='a')  
    {  
        echo "La variable SI contiene la letra a";  
    }  
?>
```



La sentencia IF: Ejemplos

- Escribir un sitio Web que tome dos número y compruebe si el primero es mayor que el segundo. En caso de que lo sea dará un mensaje.

```
<?php  
    $n1 = 45;  
    $n2 = 32;  
    if ($n1>$n2)  
    {  
        echo "$n1 es mayor que $n2";  
    }  
?>
```

45 es mayor que 32

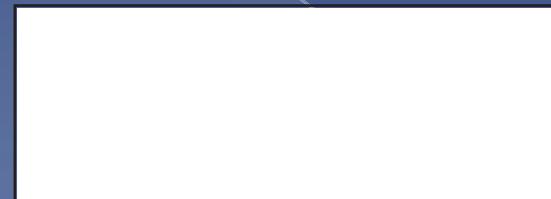
La sentencia IF: Ejemplos

- Escribir un sitio Web que tome un número y compruebe si el número es positivo.

```
<?php  
    $n1 = 45;  
    if ($n1>=0)  
    {  
        echo "$n1 positivo";  
    }  
?>
```

45 positivo

```
<?php  
    $n1 = -3;  
    if ($n1>=0)  
    {  
        echo "$n1 positivo";  
    }  
?>
```



La sentencia IF-ELSE

- La sentencia IF permite también indicar qué queremos hacer si no se cumple la condición.
- Sintaxis 2:

```
If (condición)
{
    sentencias;
}
Else
{
    sentencias;
}
```

La sentencia IF-ELSE: Ejemplos

- Escribir un sitio Web que compruebe si el contenido de una variable es "a". El sitio mostrará un mensaje con la conclusión.

```
<?php  
$letra = 'a';  
if ($letra=='a')  
{  
    echo "La variable SI contiene la letra a";  
}  
else  
{  
    echo "La variable NO contiene la letra a";  
}  
?>
```

La variable SI contiene la letra a

```
<?php  
$letra = 'z';  
if ($letra=='a')  
{  
    echo "La variable SI contiene la letra a";  
}  
else  
{  
    echo "La variable NO contiene la letra a";  
}  
?>
```

La variable NO contiene la letra a

La sentencia IF-ELSE: Ejemplos

- Escribir un sitio Web que tome dos número y compruebe si el primero es o no mayor que el segundo.

```
<?php  
    $n1 = 45;  
    $n2 = 95;  
    if ($n1>$n2)  
    {  
        echo "$n1 es mayor que $n2";  
    }  
    else  
    {  
        echo "$n1 no es mayor que $n2";  
    }  
?>
```

45 no es mayor que 95

La sentencia IF-ELSE: Ejemplos

- Escribir un sitio Web que tome un número y compruebe si el número es positivo o negativo.

```
<?php  
$n1 = -5;  
if ($n1>=0)  
{  
    echo "$n1 es positivo";  
}  
else  
{  
    echo "$n1 es negativo";  
}  
?>
```

-5 es negativo

IF ↔ IF-ELSE: Ejercicios

- Crear una web que tome dos letras y muestre cual iría primero alfabéticamente.
- Elabora un programa que contenga la edad y los nombres de 2 hermanos y muestre un mensaje indicando la edad y el nombre del mayor y además cuantos años de diferencia tiene con el menor.

IF ↔ IF-ELSE: Ejercicios

- Escribir un programa para averiguar si un usuario es mayor de edad o no.
- Escribir un programa que nos pida un número. Sacar un mensaje por pantalla indicando si el valor introducido es par o impar.

La sentencia IF-ESEIF-ELSE

- Sintaxis 3:

```
If (condición)
{
    sentencias;
}
Elseif (condición)
{
    sentencias;
}
Else
{
    sentencias
}
```

IF → IF-ELSEIF → ELSE : Ejercicios

- Escribir un programa que tome dos números y muestre qué número es mayor. Deberá además tener en cuenta si los dos números son iguales.
- Escribir un programa que nos pida un número. Sacar un mensaje por pantalla indicando si el valor introducido es positivo, negativo o cero.

IF → IF-ELSEIF → ELSE : Ejercicios

- Escribir un programa que tome dos números y muestre el resultado de restar los dos números. Es imprescindible que el resultado sea positivo y en caso de que los dos número sean iguales mostrar un aviso indicándolo.

IF → IF-ELSEIF → ELSE : Ejercicios

- Escribir un programa que tome dos palabras y las muestre por pantalla ordenadas por orden alfabético. Mostrar un aviso de error si las dos palabras son iguales.
- Escribir un programa que tome 3 números y los muestre ordenados de mayor a menor.



Programación en PHP

Estructuras iterativas



INTRODUCCIÓN

Introducción

- Normalmente las sentencias de un programa se ejecutan una a una y además sólo **una vez** cada una de ellas.

Introducción

- Normalmente las sentencias de un programa se ejecutan una a una y además sólo **una vez** cada una de ellas.
- Sin embargo, en ocasiones necesitaríamos que una o varias sentencias se ejecuten **varias veces**.

Introducción

- Normalmente las sentencias de un programa se ejecutan una a una y además sólo **una vez** cada una de ellas.
- Sin embargo, en ocasiones necesitaríamos que una o varias sentencias se ejecuten **varias veces**.
- En PHP existen 3 estructuras que permiten que una serie de sentencias se repitan hasta que se cumpla una determinada condición.

Introducción

- Los llamados **bucles** son estructuras que permiten agrupar una serie de sentencias y repetirlas una y otra vez.

Introducción

- Los llamados **bucles** son estructuras que permiten agrupar una serie de sentencias y repetirlas una y otra vez.
- Todo bucle debe incluir una **condición**.

Introducción

- Los llamados **bucles** son estructuras que permiten agrupar una serie de sentencias y repetirlas una y otra vez.
- Todo bucle debe incluir una **condición**.
- La condición del bucle indica cuándo debe parar de repetirse.

Introducción

- Los bucles que existen en PHP son:

Introducción

- Los bucles que existen en PHP son:
 - Do – While: repite ... mientras...

Introducción

- Los bucles que existen en PHP son:
 - Do – While: repite ... mientras...
 - While: mientras...

Introducción

- Los bucles que existen en PHP son:
 - Do – While: repite ... mientras...
 - While: mientras...
 - For: para...

Introducción

- Los bucles que existen en PHP son:
 - Do – While: repite ... mientras...
 - While: mientras...
 - For: para...
 - Foreach: para cada elemento...



EL BUCLE DO – WHILE

El bucle DO...WHILE

- Trabaja con la filosofía de “*repite todo esto mientras que se cumpla la condición*”.

El bucle DO...WHILE

- Trabaja con la filosofía de “*repite todo esto mientras que se cumpla la condición*”.
- Primero se ejecuta el conjunto de sentencias una vez.

El bucle DO...WHILE

- Trabaja con la filosofía de “*repite todo esto mientras que se cumpla la condición*”.
- Primero se ejecuta el conjunto de sentencias una vez.
- Despues se comprueba la condición.

El bucle DO...WHILE

- Trabaja con la filosofía de “*repite todo esto mientras que se cumpla la condición*”.
- Primero se ejecuta el conjunto de sentencias una vez.
- Después se comprueba la condición.
 - Si la condición **se cumple**, el bucle vuelve a ejecutarse

El bucle DO...WHILE

- Trabaja con la filosofía de “*repite todo esto mientras que se cumpla la condición*”.
- Primero se ejecuta el conjunto de sentencias una vez.
- Después se comprueba la condición.
 - Si la condición **se cumple**, el bucle vuelve a ejecutarse
 - Si la condición **no se cumple**, el bucle termina y el programa sigue su curso.

El bucle DO...WHILE

- Trabaja con la filosofía de “*repite todo esto mientras que se cumpla la condición*”.
- Primero se ejecuta el conjunto de sentencias una vez.
- Después se comprueba la condición.
 - Si la condición **se cumple**, el bucle vuelve a ejecutarse
 - Si la condición **no se cumple**, el bucle termina y el programa sigue su curso.
- Siempre se ejecuta al menos **una vez**.

El bucle DO...WHILE

- Sintaxis:

```
do  
{  
    sentencias;  
    ...  
}while (condición);
```

El bucle DO...WHILE

- La condición que debe comprobar el bucle suelen ser del tipo:

El bucle DO...WHILE

- La condición que debe comprobar el bucle suelen ser del tipo:
 - Variable > constante
 - Variable \geq constante
 - Variable < constante
 - Variable \leq constante

El bucle DO...WHILE

- La condición que debe comprobar el bucle suelen ser del tipo:
 - Variable > constante
 - Variable \geq constante
 - Variable < constante
 - Variable \leq constante
- Existe la posibilidad de comprobar varias condiciones uniéndolas utilizando **and** y **or** (**&&** - **||**)



EJERCICIOS

DO - WHILE

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP en la que se muestren los 10 primeros números.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que tome dos números cualesquiera y muestre por pantalla los números que hay en medio de ambos números.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que cree un array de 5 posiciones y lo muestre por pantalla.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que muestre todos los números pares que hay entre 50 y 100.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que muestre pantalla la suma de los números entre 1 y 15.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que muestre los múltiplos de 3 que hay entre 1 y 100.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que tenga un vector de 10 posiciones y muestre por pantalla sólo los valores mayores que 30.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que tome un vector de 15 números y muestre por pantalla sólo los números pares. Además deberá decir cuántos números impares hay.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que tome un array de 10 valores y muestre por pantalla la suma de todos los valores.

Ejercicios DO – WHILE

- ✓ Escribir una página en PHP que tome un array de 10 valores y muestre por pantalla sólo los valores que están en una posición par.



Programación en PHP

Estructuras de control e iterativas



Programación en PHP

Estructuras iterativas



EL BUCLE WHILE

El bucle WHILE

- Trabaja con la filosofía de “*mientras se cumpla la condición repite todo esto*”.
- Primero se comprueba la condición.
 - Si la condición se cumple, el bucle empieza a ejecutarse
 - Si la condición no se cumple, el bucle no se ejecuta y el programa sigue su curso.
- Es posible que no se ejecute ninguna vez.

El bucle WHILE

- Sintaxis:

```
while (condición)
{
    sentencias;
}
```

Ejercicios WHILE

- ✓ Escribir una página en PHP en la que se muestren los 10 primeros números.
- ✓ Escribir una página en PHP que calcule el factorial de 15.
- ✓ Escribir una página en PHP que tome dos números cualesquiera y muestre por pantalla los números que hay en medio de ambos números.

Ejercicios WHILE

- ✓ Escribir una página en PHP que muestre todos los números pares que hay entre 50 y 100.

- ✓ Escribir una página en PHP que muestre pantalla la suma de los números entre 1 y 15.

- ✓ Escribir una página en PHP que muestre los múltiplos de 3 que hay entre 1 y 100.

Ejercicios WHILE

- ✓ Escribir una página en PHP que tenga un vector de 10 posiciones y lo muestre por pantalla.
- ✓ Escribir una página en PHP que tome un vector de 15 números y muestre por pantalla sólo los números pares. Además deberá decir cuántos números impares hay.

Ejercicios WHILE

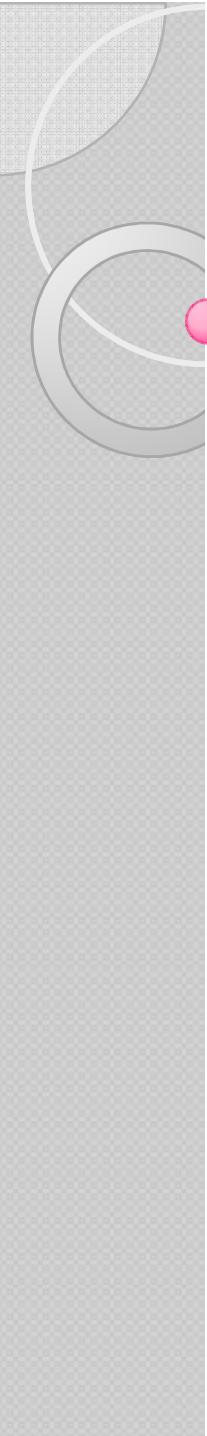
- ✓ Escribir una página en PHP que tome un array de 10 valores y muestre por pantalla la suma de todos los valores.

- ✓ Escribir una página en PHP que tome un array de 10 valores y muestre por pantalla sólo los valores que están en una posición par.



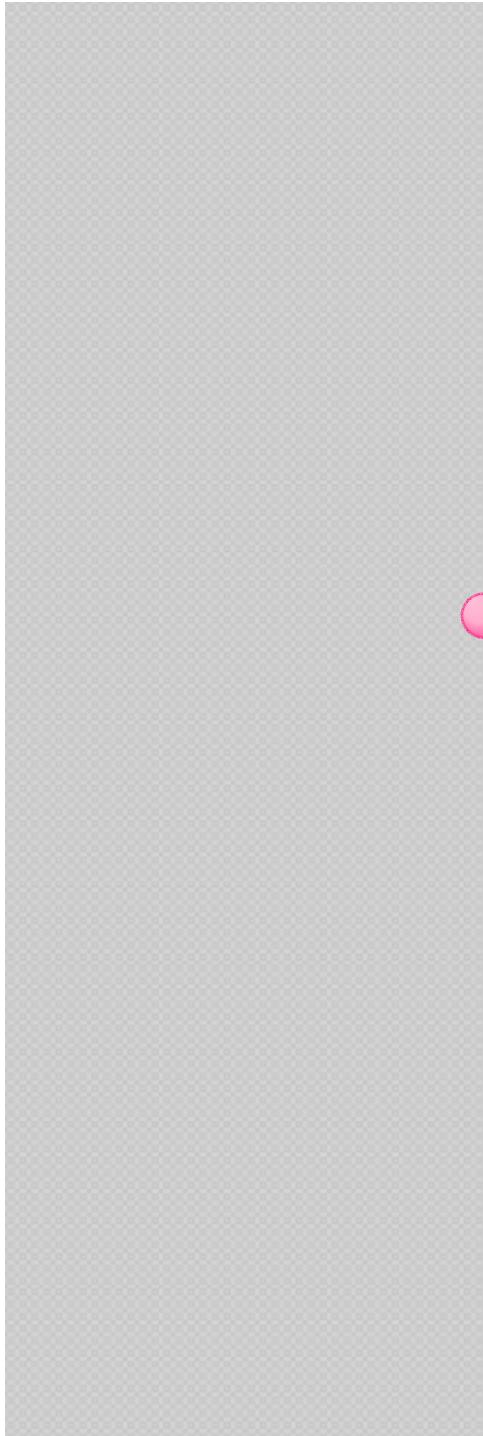
Programación en PHP

Estructuras de control e iterativas

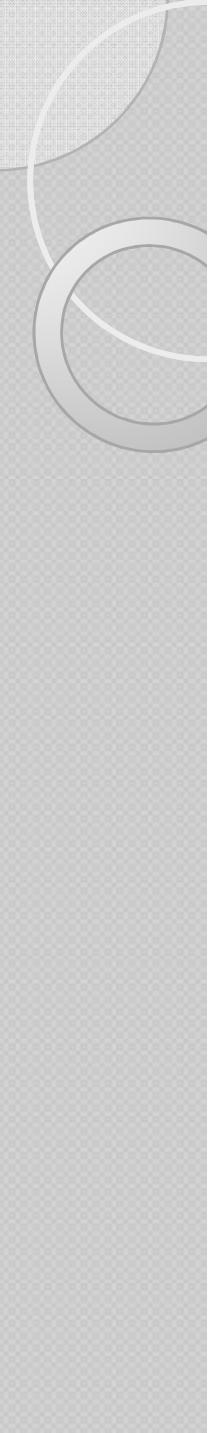


Programación en PHP

Estructuras iterativas

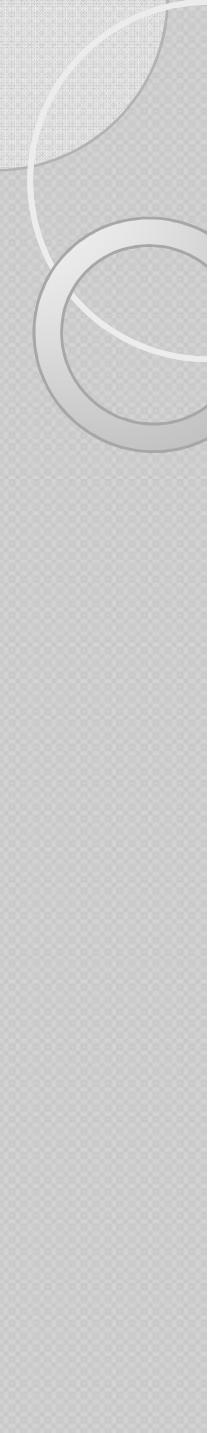


• EL BUCLE FOR



El bucle FOR

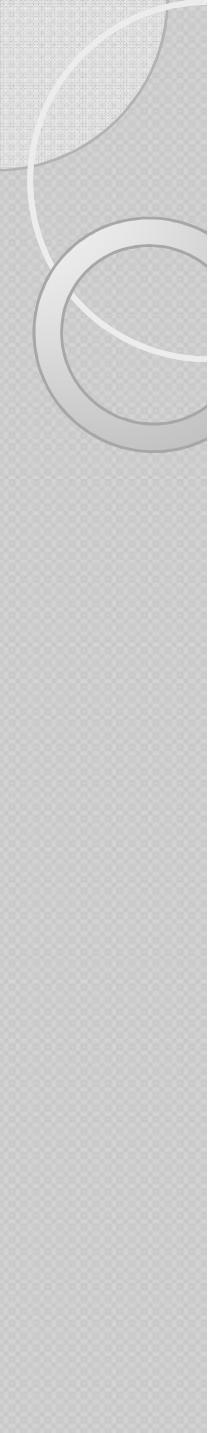
- Aunque el bucle FOR tiene la misma filosofía que el resto (repetir un conjunto de sentencias) su sintaxis es mucho más reducida.
- En la propia cabecera del bucle se indica:
 - Desde dónde empezamos.
 - Condición para parar.
 - Variación en la variable contador.



El bucle FOR

- Sintaxis:

```
for(inicio; condición; incremento)
{
    sentencias;
}
```

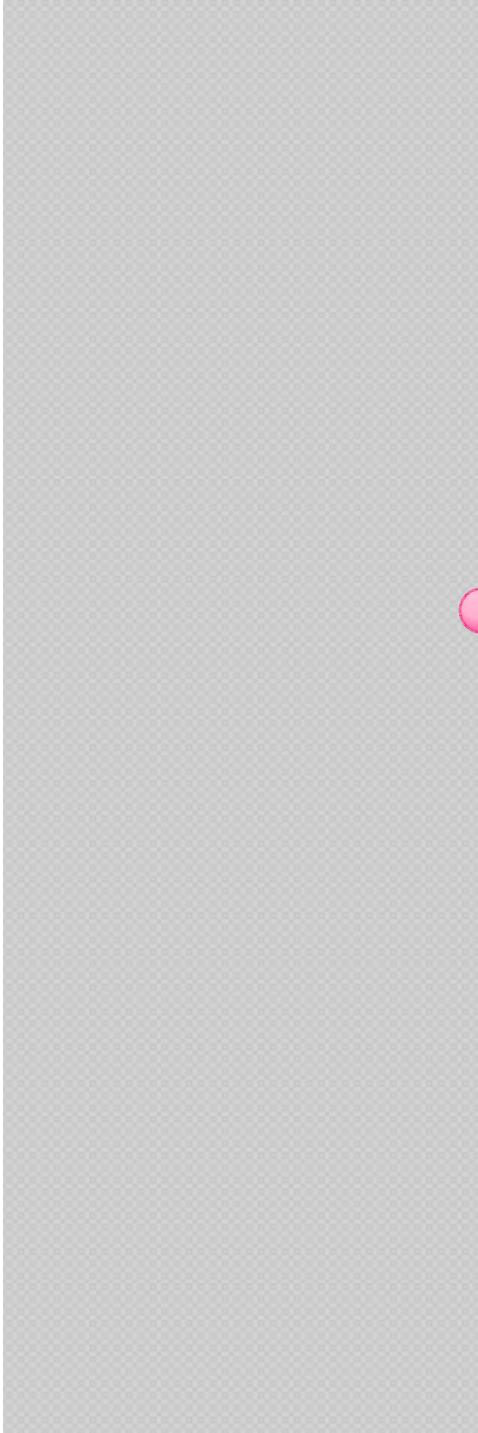


Ejercicios FOR

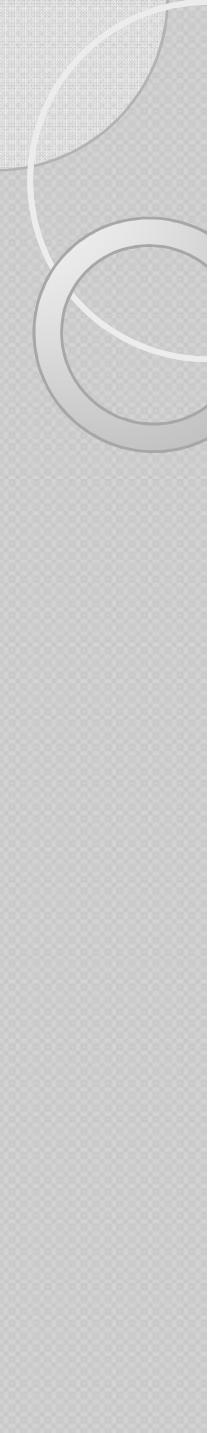
- ✓ Escribir una página en PHP en la que se muestren los 10 primeros números.

- ✓ Escribir una página en PHP que calcule el factorial de 15.

- ✓ Escribir una página en PHP que tome dos números cualesquiera y muestre por pantalla los números que hay en medio de ambos números.



- ANIDAR BUCLES



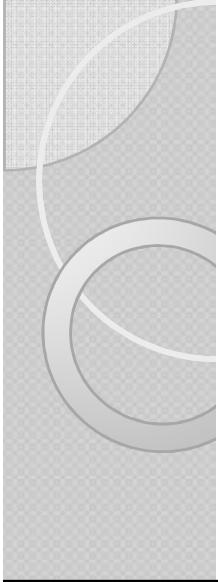
Anidar bucles

- Una de las ventajas que aportan los bucles (cualquiera de los tres) es que dentro pueden tener cualquier otra sentencia.
- Esto incluye **otros bucles**.
- Bucle anidados.
- Permite repetir un bucle **completo** tantas veces como queramos.

Ejemplo

- Mostrar el contenido de un vector de 5 posiciones 10 veces.

```
<?php  
    $vector = array (5, 9, 8, 4, 9);  
  
    for ($i=0; $i<10; $i++)  
    {  
        for ($j=0; $j<5; $j++)  
        {  
            echo $vector[$j]. " ";  
        }  
        echo "<br>";  
    }  
?>
```

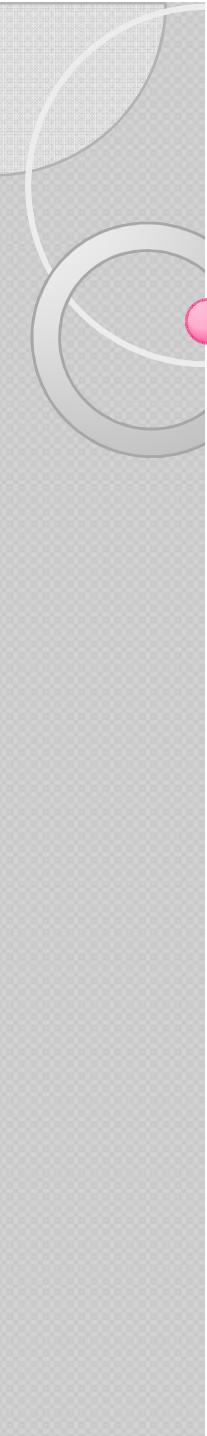


Ejemplo 2

- Crear una matriz de 3 filas y 2 columnas.
Mostrar el contenido de la matriz en forma de tabla.

```
echo "<table border>";
for ($i=0; $i<3; $i++)
{
    echo "<tr>";
    for($j=0; $j<2; $j++)
    {
        echo "<td>".$matrix[$i][$j]."</td>";
    }
    echo "</tr>";
}
echo "</table>";
```

\$matrix[0][0] = 4;	\$matrix[0][1] = 9;
\$matrix[1][0] = 12;	\$matrix[1][1] = 15;
\$matrix[2][0] = -10;	\$matrix[2][1] = 58;



Programación en PHP

Estructuras de control e iterativas

ARRAYS ASOCIATIVOS

Programando PHP

Introducción

Arrays asociativos

- ▶ Las posiciones de los arrays asociativos no están numeradas, si no que tienen un nombre.

Valor	León	Seat		2.500	V6
Posición	modelo	marca	fecha	Cc	Motor



Arrays asociativos

- ▶ La declaración de un array asociativo como si cada posición fuera independiente.
- ▶ Se indica en cada caso cómo se llamará la posición.
- ▶ El acceso a las distintas posiciones de un array asociativo se hace de indicando entre [] el nombre de la posición a la que se quiere acceder.



Arrays asociativos: Asignación

```
<?php
```

```
    $matriz2 ['modelo'] = "León";  
    $matriz2 ['marca'] = "Seat";  
    $matriz2 ['fecha'] = null;  
    $matriz2 ['cc'] = "2.500";  
    $matriz2 ['motor'] = "V6";
```

```
?>
```



Ejercicios

Ejercicios

- ▶ Crear el siguiente array asociativo y mostrarlo dentro de una tabla igual a la siguiente:

Nombre	Apellido	Edad	Altura	Peso	Pelo	Estado
Jorge	Pérez	35	1,77	80	Moreno	Soltero



```
<?php  
$persona['nombre'] = 'Jorge';  
$persona['apellido'] = 'Pérez';  
$persona['edad'] = 35;  
$persona['altura'] = 1.77;  
$persona['peso'] = 80;  
$persona['pelo'] = 'Moreno';  
$persona['estado']= 'Soltero';  
  
echo "<table border>";  
echo "<tr bgcolor='66ff33'>";  
echo "<td> Nombre </td><td>Apellido</td><td>Edad</td>";  
echo "<td>Altura</td><td>Peso</td><td>Pelo</td><td>Estado</td>";  
echo "</tr>";  
echo "<tr bgcolor='#C1FFD1'>";  
echo "<td>".$persona['nombre']."</td>";  
echo "<td>".$persona['apellido']."</td>";  
echo "<td>".$persona['edad']."</td>";  
echo "<td>".$persona['altura']."</td>";  
echo "<td>".$persona['peso']."</td>";  
echo "<td>".$persona['pelo']."</td>";  
echo "<td>".$persona['estado']."</td>";  
echo "</tr></table>";  
?>
```

Ejercicios

- ▶ Crear un array asociativo en el que el nombre de las posiciones sean los meses del año. El contenido de cada posición será el número de días del mes. Mostrar por pantalla la siguiente lista:
 - Enero: 31 días
 - Febrero: 28 días
 - Marzo: 31 días
 - ...



Ejercicios

- ▶ Crear un array asociativo que almacene las notas de un alumno. El nombre de cada posición será las siglas de la asignatura y el contenido será la nota que tiene el alumno en esa asignatura.
 - ▶ Mostrar por pantalla todas las notas del alumno
 - ▶ Mostrar por pantalla la nota media de expediente del alumno.





La función ARRAY

Creación de arrays asociativos

- ▶ Los arrays asociativos se pueden crear creando uno a uno todos sus elementos.
- ▶ También se puede usar la función **array**
- ▶ Habrá que indicar siempre posición => contenido

```
$persona['nombre'] = 'Jorge';  
$persona['apellido'] = 'Pérez';  
$persona['edad'] = 35;  
...  
  
$persona = array ('nombre'=>'Jorge', 'apellido'=>'Pérez',  
                  'edad'=>35, 'altura'=>1.77, 'peso'=>80  
                  'pelo'=>'moreno', 'estado'=>'soltero');
```



Recorrer arrays asociativos

Recorrer arrays asociativos

- ▶ Posiciones no numéricas → no podemos usar bucles
 - ▶ FOR
 - ▶ DO-WHILE
 - ▶ WHILE
- ▶ En PHP existe un bucle creado principalmente para recorrer los arrays asociativos.
 - ▶ FOREACH



El bucle Foreach

► Sintaxis

```
Foreach ($array as $variable)
{
    Sentencias;
}
```

► Funcionamiento:

- ▶ En cada vuelta, el propio bucle copiará un valor del array en la variable simple.
- ▶ A partir de ese momento, se puede usar la variable simple para lo que se necesite.



Ejercicios

Ejercicios

- ▶ Crear un array asociativo de 5 posiciones y mostrarlo por pantalla usando el bucle FOREACH

```
<?php  
$v1 = array ('saludo'=>'Hola', 'nombre'=>'Pedro',  
             'apellido'=>'Fernández', 'edad'=>29,  
             'telefono'=>'598566955');  
  
foreach ($v1 as $a)  
{  
    echo "$a  ";  
}  
?>
```

```
<?php  
$v1['saludo'] = 'Hola';  
$v1['nombre'] = 'Pedro';  
$v1['apellido'] = 'Fernández';  
$v1['edad'] = 29;  
$v1['telefono'] = '598566955';  
  
foreach ($v1 as $a)  
{  
    echo "$a  ";  
}  
?>
```



Ejercicio

- ▶ Crear un array asociativo que almacene los datos de una mascota y mostrarlo de la siguiente forma:

Nombre	Peso	Edad	Color	Raza
Rufi	2	5	Blanco	Dálmata



?

Ejercicio

- ▶ Crear un array asociativo que tenga el número de cartulinas de colores que hay en un almacén.

Azules	Verdes	Rojos	Amarillo	Naranjas
5	9	7	25	9

- ▶ Utilizando bucles, mostrarlo dentro de una tabla como se ve arriba.
- ▶ Además, se deberá mostrar el total de cartulinas que hay.
- ▶ NOTA: se debe respetar el estilo de la tabla

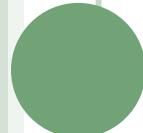
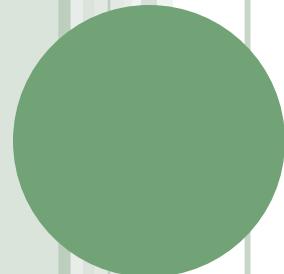


```
<?php  
    $cartulinas = array ("azules"=>5, "verdes"=>9,  
    "rojas"=>7, "amarillas"=>25, "naranjas"=>9);  
    $suma = 0;  
  
    echo "<table border>";  
    echo "<tr> <td> azules </td>";  
    echo "<td> verdes </td>";  
    echo "<td> rojas </td>";  
    echo "<td> amarillas </td>";  
    echo "<td> naranjas</td></tr>";  
    echo "<tr>";  
  
    foreach ($cartulinas as $una)  
    {  
        echo "<td> $una </td>";  
        $suma+=$una;  
    }  
    echo "</tr></table>";  
    echo "En total hay $suma cartulinas";  
?  
?
```



ARRAYS ASOCIATIVOS

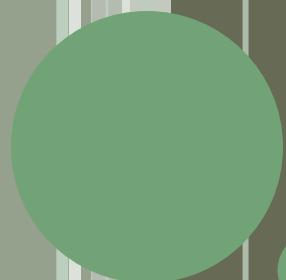
Programando PHP



ARRAYS

Programando en PHP





UN PEQUEÑO INCISO

¿QUÉ ES UNA FUNCIÓN?

- Una función es un **trozo de código** que se almacena bajo un **nombre**.
- Cuando se llama a una función, el código almacenado se ejecuta.
- De una función nos interesa lo que hace, pero NO cómo lo hace.
- Algunas funciones necesitan una serie de parámetros de entrada para funcionar.



¿QUÉ ES UNA FUNCIÓN?

- Hay dos tipos de funciones:
 - Devuelven datos
 - No devuelven datos
- Cuando una función devuelve un dato, dicho dato hay que recogerlo en una variable.
- Una función se puede llamar tantas veces como sea necesario.



¿QUÉ ES UNA FUNCIÓN?

- Las llamadas a una función se hacen de la siguiente forma:

Variable_recogida = nombre_funcion (parametros)

Nombre_funcion (parametros)

- En PHP existen muchísimas funciones predefinidas que nos ayudan a realizar las tareas.

- **Php.net**





INTRODUCCIÓN

Introducción



INTRODUCCIÓN

- Los arrays son una parte muy importante de cualquier lenguaje de programación.
- Permiten:
 - Manejar grupos de valores relacionados
 - Almacenar múltiples valores en una sola estructura y bajo un mismo nombre.
- Muchas de las funciones de PHP devuelven un array de valores.
- En PHP los arrays están muy ligados a las bases de datos.
- Tipos de arrays:
 - Posicionales
 - Asociativos.



Los arrays posicionales



ARRAYS POSICIONALES

- Formados por un conjunto de valores ordenados respecto a un índice.
- El índice entero, indica la posición del elemento en el conjunto.
- Formas de asignar un array:
 - La más sencilla → Asignar a cada posición el valor.
 - Utilizando la función array()

ASIGNACIÓN DE ARRAYS POSICIONALES

```
$array1[0]=12;  
$array1[1]="verde";  
$array1[2]=25.4;  
$array1[3]="vivo";  
$array1[]="Riviera";
```

```
$array2 = array (12, "verde", 25.4, "vivo", "Riviera");
```

Arrays

Posición	0	1	2	3	4
Array 1	12	verde	25.4	vivo	Riviera
Array 2	12	verde	25.4	vivo	Riviera



Los arrays asociativos



ARRAYS ASOCIATIVOS

- Formados por un conjunto de valores ordenados respecto a un índice que no es entero si no string.
- Formas de asignar un array:
 - La más sencilla → Asignar a cada posición el valor.
 - Utilizando la función array(). En este caso será necesario indicar el nombre de la posición.

ASIGNACIÓN DE ARRAYS ASOCIATIVOS

```
<?php  
    $array1["posi1"] = 12;  
    $array1["posi2"] = "verde";  
    $array1["posi3"] = 25.4;  
    $array1["posi4"] = "vivo";  
    $array1["posi5"] = "Riviera";  
  
    $array2 = array ("posi1"=>12, "posi2"=>"verde", "posi3"=>25.4,  
    ...     ...     ...     ... "posi4"=>"vivo", "posi5"=>"Riviera");
```

Arrays

Posición	0	1	2	3	4
Array 1	12	verde	25.4	vivo	Riviera
Array 2	12	verde	25.4	vivo	Riviera



Arrays multidimensionales



ARRAYS MULTIDIMENSIONALES

- PHP nos permite definir arrays multidimensionales mediante la combinación de arrays unidimensionales (tanto posicionales como asociativos)
- Veamos ejemplos:



ASIGNACIÓN DE ARRAYS MULTIDIMENSIONALES

```
<?php  
$matriz1[0][0] = "Peseta";  
$matriz1[0][1] = 166.386;  
$matriz1[1][0] = "Dólar";  
$matriz1[1][1] = 0.96;  
  
$matriz2[0] = array ("Peseta", 166.386);  
$matriz2[1] = array ("Dólar", 0.96);  
  
$matriz3 = array (array ("Peseta", 166.386),  
                 array ("Dólar", 0.96));
```

matrices

	Moneda	Cambio €
\$matriz1[0]	Peseta	166.386
\$matriz1[1]	Dólar	0.96
\$matriz2[0]	Peseta	166.386
\$matriz2[1]	Dólar	0.96
\$matriz3[0]	Peseta	166.386
\$matriz3[1]	Dólar	0.96

Recorrer arrays posicionales



RECORRER ARRAYS POSICIONALES

- Lo más habitual cuando se trabaja con arrays es recorrerlos para obtener sus elementos.
- La forma más sencilla de hacerlo es utilizando bucles.
- Problema: Debemos conocer a priori el tamaño.
- **count (array)**
 - Devuelve el número de elementos que hay en el array.

```
for ($i=0; $i<count($array); $i++)  
{  
    echo "$array[$i]";  
}
```



```
<?php
```

```
    $a[0] = 1;  
    $a[1] = 3;  
    $a[2] = 5;  
    $result = count($a);  
    // $result == 3
```

```
    $b[0] = 7;  
    $b[5] = 9;  
    $b[10] = 11;  
    $result = count($b);  
    // $result == 3
```

```
    $result = count(null);  
    // $result == 0
```

```
    $result = count(false);  
    // $result == 1
```

```
?>
```

Recorrer arrays asociativos



RECORRER ARRAYS ASOCIATIVOS

- Además de saber el número de elementos que tiene el array, **deberíamos** saber las claves para poder acceder a ellos.
- PHP incluye una serie de funciones que nos van a hacer la vida más fácil a la hora de utilizar arrays asociativos.



RECORRER ARRAYS ASOCIATIVOS

o **array_keys(array)**

- Devuelve un nuevo array **posicional** con las claves que forman el array.

o **array_values(array)**

- Devuelve un nuevo array **posicional** con los valores que forman parte del array.



EJEMPLO RECORRER

```
$claves = array_keys($vector1);
$valores = array_values($vector1);
for($i=0; $i<count($claves); $i++)
{
    echo "<tr align='center'><td>".$claves[$i]."</td>";
    echo "<td>".$valores[$i]."</td></tr>";
}
```

Ejercicio

- Crear un array asociativo y utilizando las funciones **array_keys** y **array_values** junto con los bucles necesarios, mostrarlo de la siguiente forma:

Nombre	Altura	Edad	Pelo	Ciudad
Juan	3.5	25	Moreno	Granada



```
<?php  
    $persona = array ('nombre'=>'Juan', 'altura'=>3.5,  
                      'edad'=>25, 'pelo'=>'moreno',  
                      'ciudad'=>'granada');  
    $posiciones = array_keys ($persona);  
    $i=0;  
    echo "<table border><tr>";  
    do  
    {  
        echo "<td>";  
        echo $posiciones[$i];  
        echo "</td>";  
        $i++;  
    }while ($i<count ($posiciones));  
    echo "</tr>";  
  
    $valores = array_values ($persona);  
    echo "<tr>";  
    $i=0;  
    do  
    {  
        echo "<td>";  
        echo $valores[$i];  
        echo "</td>";  
        $i++;  
    }while ($i<count ($valores));  
    echo "</tr></table>";  
?  
?
```

Recorrer arrays asociativos

- También podemos recorrer un array asociativo utilizando el bucle foreach.
- En este caso la sintaxis cambia un poco:

```
Foreach ($array as $posicion=>$valor)  
{  
    Sentencias  
}
```



Ejercicio

- Crear un array asociativo y utilizando el bucle foreach, mostrarlo de la siguiente forma:

Nombre	Juan
Altura	3.5
Edad	25
Pelo	Moreno
Ciudad	Granada



```
<?php

$a['nombre']='Juan';
$a['altura']=3.5;
$a['edad']=25;
$a['pelo']='Moreno';
$a['ciudad']='Granada';

echo "<table border>";

foreach ($a as $posi=>$valor)
{
    echo "<tr>";
    echo "<td> $posi </td> <td> $valor </td>"; 
    echo "</tr>";
}
echo "</table>";

?>
```

Ordenar Arrays



ORDENAR ARRAYS

- o `sort(array)`

- Ordena alfabéticamente los valores.
- De menor a mayor.
- Se modifica el array original
- Se pierde la relación entre índice y valor

- o `rsort(array)`

- Ordena de forma inversa a sort.



```

<?php
    $frutas = array("limón", "naranja",
                    "platano", "albaricoque");
    sort($frutas);
    foreach ($frutas as $clave => $valor) {
        echo "<td>frutas[" . $clave . "]</td>";
        echo "<td> $valor </td>";
        echo "<tr></tr>";
    }
?>

```

Posicion	Valor
frutas[0]	albaricoque
frutas[1]	limón
frutas[2]	naranja
frutas[3]	platano

ORDENAR ARRAYS

- `asort(array)`
 - Ordena igual que `sort`, pero mantiene la relación entre índice y valor.

- `arsort(array)`
 - Ordena de forma inversa a `asort`.



EJEMPLO ORDENACIÓN

Vector sin ordenar

Posición	Valor
0	Madrid
1	Zaragoza
2	Bilbao
3	Valencia
4	Lérida
5	Alicante

Vector ordenado con sort

Posición	Valor
0	Alicante
1	Bilbao
2	Lérida
3	Madrid
4	Valencia
5	Zaragoza

Vector ordenado con asort

clave	Valor
5	Alicante
2	Bilbao
4	Lérida
0	Madrid
3	Valencia
1	Zaragoza



ORDENAR ARRAYS

- **ksort(array)**

- Ordena alfanuméricamente las claves de un array de menor a mayor.
- Mantiene las relaciones entre índice y valor.

- **krsort(array)**

- Ordena de forma inversa a ksort.



EJEMPLO ORDENACIÓN

```
$vector = array ('d'=> 'Madrid', 'c'=> 'Zaragoza',
                 'e'=> 'Bilbao', 'b'=> 'Valencia',
                 'f'=> 'Lérida', 'a'=> 'Alicante');
```

Vector sin ordenar

Posición	Valor
d	Madrid
c	Zaragoza
e	Bilbao
b	Valencia
f	Lérida
a	Alicante

Vector ordenado con ksort

Posición	Valor
a	Alicante
b	Valencia
c	Zaragoza
d	Madrid
e	Bilbao
f	Lérida



Otras funciones de arrays



OTRAS FUNCIONES DE ARRAYS

oarray_reverse (array)

- Devuelve el array pasado como parámetro, pero con sus componentes en orden inverso.



OTRAS FUNCIONES DE ARRAYS

- `array_count_values(array)`
 - Devuelve un array en el que:
 - Los índices son los contenidos del array original
 - Los valores asociados son la frecuencia de repetición de dichos valores en el array original.



```
$vector = array (1, 2, 4, 2, 5, 1, 2, 4, 2, 5, 6);  
$vector_count = array_count_values ($vector);
```

Vector resultado de count

claves	Frecuencia
1	2
2	4
4	2
5	2
6	1



OTRAS FUNCIONES DE ARRAYS

- `in_array(elemento_buscar, array)`
 - Nos dice si un elemento está dentro de un array o no.
- `compact()`
 - Recibe como argumento una lista de variables que han sido definidas previamente.
 - Devuelve un array en el que los índices son los nombres de las variables y el contenido, sus correspondientes valores.

Un pequeño inciso

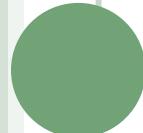
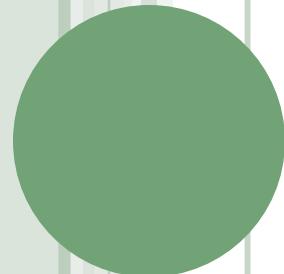


NÚMEROS ALEATORIOS

- `rand(inf, sup)` o `mt_rand(inf, sup)`
 - Ambas devuelven un número aleatorio entre el límite inferior y el límite superior.
 - En ambos casos hay que inicializar la semilla a partir de la que se van a generar los números:
 - `srand(semilla)` o `mt_srand(semilla)`
 - Esto no es necesario a partir de PHP 4.2.0 que ya lo hace de forma automática.

```
srand(8);  
$num_aleatorio = rand(1, 15);  
$num_aleatorio2= rand(1, 15);  
$num_aleatorio3= rand(10, 80);
```

```
srand(time());  
$num_aleatorio = rand(1, 15);  
$num_aleatorio2= rand(1, 15);  
$num_aleatorio3= rand(10, 80);
```



ARRAYS

Programando en PHP



MANEJO DE FORMULARIOS

Programando con PHP

Manejo de formularios

Introducción

Introducción

- Un formulario es un conjunto de elementos que permiten obtener información del usuario.
- Tipos de elementos de un formulario:
 - Campos de texto: text
 - Conjunto selección circular: radio
 - Casillas de verificación: checkbox
 - Botones: submit – reset
 - Campos de contraseña: password

Introducción

- ❑ Es muy importante recordar en este punto que cada elemento de un formulario tenía como mínimo un atributo:
 - ❑ NAME: indicaba el nombre del elemento.
- ❑ Además, podía tener más atributos:
 - ❑ VALUE: valor que contiene el elemento

Introducción

- En la propia etiqueta de <form> era necesario indicar:
 - **Action**: programa que manejará el formulario.
 - **Method**: forma en que se enviarán los datos indicados por el usuario al servidor.
 - **Enctype**: método de encriptación utilizado.
 - **Accept-charset**: juego de caracteres que acepta el servidor.

Elementos de un formulario

Elementos

- Entre las etiquetas <form> y </form>...
- Podemos introducir un área de texto con las etiquetas <textarea> y </textarea>.
- Textarea acepta los siguientes atributos:
 - Name: nombre del área de texto
 - Rows: número de líneas que se podrán visualizar
 - Cols: número de caracteres por línea.

Elementos

- El elemento más utilizado es <input> </input> que engloba a la mayoría de los elementos de entrada de datos.
- Atributos de <input>
 - Name: nombre del elemento dentro del formulario
 - Value: valor inicial del elemento
 - Checked: opción preseleccionada
 - Type: tipo de elemento:

Text	Password	Checkbox	Radio	Submit
Reset	File	Hidden	Image	Button

Elementos

- El elemento **text** permite introducir en el formulario un campo de texto.
- Tiene los siguiente parámetros:
 - **name='nombre'** Asigna un nombre único del elemento
 - **size='n'** Tamaño de la caja de texto que aparece en pantalla.
 - **maxlength='n'** Número máximo de caracteres.
 - **value=' texto '** Texto que aparecerá inicialmente.
 - **disabled** Desactiva la caja de texto, por tanto el usuario no podrá insertar ningún texto en dicho campo.
 - **readonly** El texto no puede ser modificado por el usuario.
 - **tabindex='n'** Asigna el orden de tabulador que tiene el campo respecto a los demás elementos que componen el formulario..

Elementos

- El elemento **password** permite introducir en el formulario un campo de texto pero el texto será sustituido por caracteres especiales.
- Tiene los mismos parámetros que **text**.
- El elemento **hidden** inserta en el formulario un campo escondido, por lo que no se ve en pantalla.
- Sus atributos son:
 - **name= ‘nombre’** Asigna un nombre al campo oculto.
 - **value= ‘valor’**. Asigna un valor campo.

Elementos

- El elemento **Radio** es un campo multivalor excluyente, permite escoger una y sólo una opción de un conjunto de valores.
- Posee los siguientes parámetros:
 - **name='nombre'**. Asigna un nombre al conjunto. Este identificador debe ser el mismo para todos los elementos radio de un grupo.
 - **value='valor'**. Asigna un valor a la variable de cada casilla que componen el elemento radio.
 - **checked**. Selecciona por defecto uno de las opciones del grupo
 - **disabled**. Desactiva el radio botón, por lo que el usuario no podrá marcarlo.
 - **tabindex='n'**. Especifica el orden de tabulador que tendrá el campo respecto todos los elementos que componen el formulario.

Elementos

- El elemento **Checkbox** define las casillas de verificación que tendrá nuestro formulario.
- Este tipo de elementos pueden ser activados o desactivados por el usuario al pulsar sobre la caja.
- Este elemento no es auto excluyente.
- Tiene como parámetros:
 - **name='nombre'**. Asigna un nombre a cada elemento del grupo.
 - **value='valor'**. Asigna un valor a cada casilla de verificación que componen el checkbox.
 - **checked**. Selecciona por defecto una o más de las casillas del grupo.
 - **disabled**. Desactiva la casilla de verificación, por lo que el usuario no podrá seleccionarla.
 - **tabindex='n'**. Especifica el orden de tabulador que tendrá el campo respecto todos los elementos que componen el formulario.

Elementos

- El elemento **file** permite enviar un archivo adjunto al formulario.
- Añade una caja de texto y un botón con el que se abre un explorador para adjuntar el archivo
- Algunos de sus atributos son:
 - **Name='nombre'**: Asigna un nombre al elemento.
 - **Size='n'**: anchura de la caja de texto

Elementos

- Los elementos **submit**, **button** y **reset** insertan en el formulario botones.
 - Submit: envía el formulario para que sea manejado por el programa indicado en 'action'
 - Reset: resetea el formulario dejandolo con los valores por defecto
 - Button: No hace nada. Su uso más frecuente es para la programación de eventos desde el lado del cliente.
- Sus parametros son:
 - **name='nombre'**. Asigna un nombre al botón,
 - **value='valor'**. Define el texto que veremos en el botón
 - **disabled**. Desactiva el botón, cuando se pulsa no se produce ninguna acción..
 - **tabindex ='n'**. Especifica el orden de tabulador que tendrá el campo respecto todos los elementos que incluye el formulario.

Listas desplegables

- La etiqueta **<select>** crea una lista desplegable con varias opciones.
- Atributos de select:
 - **Name='nombre'**: nombre de la lista desplegable
 - **Size='n'**: Número de filas visibles en la lista
 - **Disable**: Desactiva la lista y no se podrá elegir ninguna opción
- Entre **<select>** y **</select>** aparecerán las etiquetas **<option>** y **</option>** que marcan cada opción de la lista.
 - **Selected**: Especifica la opción predeterminada
 - **Value='valor'**: Especifica el valor del elemento elegido

Listas desplegables

```
<form name = 'formulario1'>
  <select name='ciudades'>
    <option value=0 selected> selecciona tu ciudad de nacimiento </option>
    <option value='gra'> Granada </option>
    <option value='sev'> Sevilla </option>
    <option value='mal'> Málaga </option>
    <option value='alm'> Almería </option>
    <option value='cor'> Córdoba </option>
    <option value='jae'> Jaén </option>
    <option value='hue'> Huelva </option>
    <option value='cad'> Cádiz </option>
  </select>
</form>
```

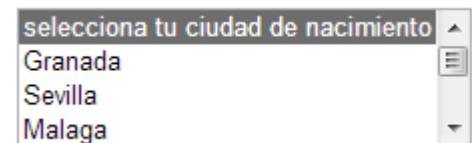
A screenshot of a dropdown menu. The title "selecciona tu ciudad de nacimiento" is at the top. Below it is a list of cities: Granada, Sevilla, Málaga, Almería, Córdoba, Jaén, Huelva, and Cádiz. The menu has a light blue background and a white border.

selecciona tu ciudad de nacimiento
selecciona tu ciudad de nacimiento
Granada
Sevilla
Málaga
Almería
Córdoba
Jaén
Huelva
Cádiz

Listas desplegables

- Utilizando el atributo 'size' podemos indicar el número de líneas que queremos visualizar desde el principio:

```
<form name = 'formulario1'>
  <select name='ciudades' size=3>
    <option value=0 selected> selecciona tu ciudad de nacimiento </option>
    <option value='gra'> Granada </option>
    <option value='sev'> Sevilla </option>
    <option value='mal'> Málaga </option>
    <option value='alm'> Almería </option>
    <option value='cor'> Córdoba </option>
    <option value='jae'> Jaén </option>
    <option value='hue'> Huelva </option>
    <option value='cad'> Cádiz </option>
  </select>
</form>
```



A screenshot of a dropdown menu window. The title bar says "selecciona tu ciudad de nacimiento". The menu contains three items: "Granada", "Sevilla", and "Málaga". There are scroll bars on the right side of the window.

Manejo de formularios

Envío de formularios al servidor

Envío de formularios

- El atributo ‘method’ especifica la forma en que el protocolo HTTP enviará los datos del formulario al servidor.
- Puede tener dos valores:
 - Get
 - El conjunto de datos se agrega a la URL especificada en ‘action’
 - Se utiliza el carácter ‘?’ como separador.
 - Post
 - Los datos se incluyen en el cuerpo del mensaje enviado.

Envío de formularios

- El método ‘get’ debe usarse cuando el formulario **no** va a tener ‘efectos secundarios’
- Muchas búsquedas en bases de datos no tienen efectos secundarios visibles → son ideales para el método ‘get’
- Si el formulario va a modificar la base de datos → debe utilizarse el método ‘post’
- Si el formulario va a incluir ficheros → debe utilizarse el método ‘post’

Envío de formularios

- ¿Qué pasa cuando el usuario envía el formulario (pulsa el botón ‘submit’)?
- Se genera de forma automática un array asociativo que tiene:
 - ▣ Posición: Nombre del elemento del formulario
 - ▣ Contenido: Valor de dicho elemento.

Ejemplo

Formulario

```
<form name = 'form' method = 'get' action = 'prueba.php'>
<input name = 'nombre' type='text'></td>
<input name = 'edad' type='text'></td>
<input name='enviar' type = 'submit'>
</form>
```

Juan

35

Enviar

Array enviado

Nombre	Edad	Enviar
'Juan'	35	'Enviar'

Envío de formularios

- Una vez enviado el formulario, se activa el programa que gestiona dicho formulario:
 - Se envía un **e-mail** si
action = 'mailto:jjj@llkjo.com'
 - Se ejecuta **PHP** si
action = 'documento.php'
 - Se ejecuta **el mismo documento** donde
está el formulario si
action='#'

Envío de formularios

- ¿Cómo se accede al array generado por un formulario?
- Dependiendo del método de envío que hayamos utilizado se generan dos arrays distintos:

ARRAY	CONTENIDO
_POST	Array con las variables pasadas a través del método POST
_GET	Array con las variables pasadas a través del método GET

- Puntualmente también se podría usar un array genérico:

ARRAY	CONTENIDO
_REQUEST	Array con las variables pasadas a través del método POST y GET

Manejo de formularios

Ejemplo de procesamiento con GET

```
<form name = 'contacto' method = 'get' action = 'formulario1.php'>
    Nombre
    <input name = 'nombre' type='text'>
    Apellidos
    <input name = 'apellidos' type='text'>
    Correo electrónico
    <input name = 'email' type='text'>
    ¿Qué apartado te ha gustado más?</td>
    <select name="seccion">
        <option value = '0' selected>--- Elige sección ---</option>
        <option value='presentacion'>Presentación</option>
        <option value='intro'>Introducción</option>
        <option value='texto'>Formato de texto</option>
        <option value='enlaces'>Hiperenlaces</option>
        <option value='imagenes'>Imagenes</option>
    </select>
    ¿Desea suscribirse a nuestro servicio de noticias?
    Si <input name='suscripcion' type = 'radio' value = 'si' checked>
    No <input name='suscripcion' type = 'radio' value = 'no'>
    Escriba aquí su consulta
    <textarea name = 'consulta' rows='10' cols='30'>Consulta...</textarea>
    <input name='enviar' type = 'submit'>
    <input name='cancelar' type = 'reset'>
</form>
```

Ejemplo de manejo (Documento php)

```
<?php  
    $nombre = $_GET["nombre"];  
    $apellidos = $_GET["apellidos"];  
    $mail = $_GET["email"];  
    $apartado = $_GET["seccion"];  
    $suscribirse = $_GET["suscripcion"];  
    $consul = $_GET["consulta"];
```

Ejemplo de manejo (Documento php)

```
if ($nombre)
{
    echo "Hola $nombre";
}

if ($apellidos)
{
    echo "$apellidos.<br>";
}
```

Ejemplo de manejo (Documento php)

```
switch ($apartado)
{
    case "intro":
        echo "Sección preferida: Introducción.";
        break;

    case "presentacion":
        echo "Sección preferida: Presentación.";
        break;

    case "texto":
        echo "Sección preferida: Formato de texto.";
        break;

    case "enlaces":
        echo "Sección preferida: Enlaces.";
        break;

    case "imagenes":
        echo "Sección preferida: Imágenes.";
        break;

    default:
        echo "No tienes sección preferida.";
}
```

Ejemplo de manejo (Documento php)

```
if ($mail and $suscribirse=='si')
{
    echo "<br><br>Has sido suscrito a nuestro canal de noticias";
    echo "<br> Dirección de suscripción: $mail<br><br>";
}

if ($consul != "Consulta...")
{

    echo "<br>Tu consulta: $consul <br>será pasada a nuestros expertos";
    echo "y esperamos responderle a la mayor brevedad posible";
    if ($mail)
    {
        echo "<br> La respuesta la enviaremos a la dirección de correo: $mail";
    }
    else
    {
        echo "<br> Sería necesario una forma de ponernos en contacto con";
        echo " usted para poder enviarle la respuesta";
    }
}
```

Nombre

Apellidos

Correo electrónico

¿Qué apartado te ha gustado más?

--- Elige sección --- ▾

¿Desea suscribirse a nuestro servicio de noticias?

Si No

Escriba aquí su consulta

Consulta...

Enviar

Restablecer

Nombre

chari

Apellidos

vargas

Correo electrónico

eag.chari@gmail.com

¿Qué apartado te ha gustado más?

Introducción ▾

¿Desea suscribirse a nuestro servicio de noticias?

Si No

Escriba aquí su consulta

¿Cómo se usan los enlaces?

Enviar

Restablecer

Ejemplo de manejo (Ejecución)

Hola chari vargas.

Sección preferida: Introducción.

Has sido suscrito a nuestro canal de noticias

Dirección de suscripción: eag.chari@gmail.com

Tu consulta: ¿Cómo se usan los enlaces?

será pasada a nuestros expertos y esperamos responderte a la mayor brevedad posible

La respuesta la enviaremos a la dirección de correo: eag.chari@gmail.com

Muchas gracias por visitar mi página

Ejercicios

Ejercicios

- Programar una página en HTML – PHP que a través de formularios pida al usuario su nombre y le muestre un saludo personalizado.
- Programar una página en HTML – PHP que a través de formularios pida al usuario dos números y muestre en una tabla el resultado de:
 - Sumarlos
 - Restarlos
 - Multiplicarlos
 - Dividirlos

ALGUNOS INCISOS

Manejo de Formularios

Envío de archivos

Utilización de <input file>

Ficheros adjuntos

- Si queremos que se pueda subir un archivo al servidor, debemos añadir obligatoriamente al formulario el tipo de encritación:
 - enctype="multipart/form-data"
- Además, el metodo de envío no podrá ser GET, deberá ser **POST**
- Dentro del formulario utilizaremos el tipo file.
 - Añade un botón con el que se abre un explorador para añadir el archivo.

Ficheros adjuntos

- En el documento .php se genera un nuevo array (tipo matriz) llamado “_FILES” que contiene una línea por cada elemento tipo file del formulario
- Cada fila de la matriz tiene las siguientes posiciones:
 - Name: nombre del archivo en el ordenador cliente
 - Type: tipo de archivo
 - Tmp_name: nombre temporal del archivo en el servidor
 - Error: true o false (1 o 0)
 - Size: tamaño en bytes del archivo subido

Ficheros adjuntos

- El campo type, contendrá los siguientes tipos de archivo:
 - Imagen en jpg: image/jpeg
 - Imagen en png: image/png
 - Imagen en gif: image/gif
 - Documentos pdf: application/pdf
 - Documentos Word 97 – 2003: application/msword
 - Documentos contexto plano (.txt): text/plain
 - ...

Ficheros adjuntos

- `move_uploaded_file($origen, $destino);`
 - ▣ Esta función mueve un archivo subido a una nueva ubicación.
 - ▣ `$origen`: nombre temporal del archivo subido
 - ▣ `$destino`: ruta completa, o relativa, de la nueva ubicación del archivo.
- `file_exists($path);`
 - ▣ Comprueba si un directorio existe o no.
 - ▣ Devuelve true o false
- `mkdir($path);`
 - ▣ Crea una carpeta en la ruta especificada.

Importante: el directorio donde se vaya a colocar el archivo debe existir

Ejemplo “Envío de archivos”

Formulario de envío

```
<html>
  <body>
    <form action="procesar.php" method="post" enctype="multipart/form-data">
      Archivo
      <input type="file" name="foto"/>
      <input type="submit" name="button" value="Enviar" />
    </form>
  </body>
</html>
```

\$_FILES['foto']

name	type	tmp_name	error	size
perro.jpg	image/jpeg	c:\wamp\tmp\php3514.tmp	0	55370

Procesamiento del documento

```
<html>
<body>

<?php
    $name = $_FILES['foto']['name']; // nombre original del archivo
    $temp = $_FILES['foto']['tmp_name']; //nombre temporal del archivo en el servidor

    // Comprobamos si existe la carpeta donde queremos guardar la foto
    if(!file_exists("./img"))
    {
        mkdir("./img");
    }

    // Guardamos el archivo en el directorio del servidor
    $ruta = "img/$name";
    move_uploaded_file ($temp,"$ruta");

?>

" />
</body>
</html>
```

Ejercicios propuestos

- Realizar un documento en HTML – PHP que solicite al usuario una imagen y un nombre para la imagen (no tiene por qué coincidir con el nombre original). El documento deberá guardar la imagen con el nuevo nombre y mostrar por pantalla la información de la imagen subida por el usuario en una tabla como la siguiente:

Nombre original	Nuevo nombre	Tamaño en megas	Tipo de imagen
...

Formularios internos

Procesar un formulario en el mismo documento

Formularios internos

- En ocasiones necesitaremos procesar un formulario en la misma página en la que lo hemos escrito.
- Si en el atributo **action** de un formulario ponemos **#**, el formulario será tratado en el mismo documento.
- Habrá que comprobar cómo se ha ejecutado el documento, es decir, si viene o no de una ejecución de un formulario.
- La función **isset** comprueba si una variable cualquiera existe o no.

Formularios internos

```
<?php
if(isset($_POST['enviar']))
{
    procesamiento.....
}

?>
<form method="post" action="#">
    nombre <input type="text" name="name"><br>
    elementos ...
        <input type="submit" name="enviar"><br>
    .....
</form>
```

Información extra

Uso de header

- header (\$cadena)
 - ▣ La función header de php nos permite enviar encabezados HTTP sin formato.
 - ▣ Uno de los encabezados más utilizados (y que a nosotros nos viene mejor) es el encabezado Location.
 - ▣ Con location redireccionamos al cliente a la página que queramos

header ("location: www.google.es")

Uso de header

- Es necesario tener en cuenta que:
 - Una vez que php lea la función header, dejará sin ejecutar todo lo que haya por debajo.
- También podemos usar header con un retraso, en este caso utilizaremos la cabecera refresh junto con la URL a la que queramos redirigir:
 - En refresh indicamos los segundos de espera antes de redirigir a la URL elegida

```
header("Refresh: 5; URL= http://www.hotmail.com");
```

MANEJO DE FORMULARIOS

Programando con PHP

Manejo de Cadenas

Programando con PHP

Introducción

- El tratamiento de cadenas es muy importante en PHP.
- El control de usuarios y contraseñas está muy ligado al tratamiento de cadenas.
- Existe un amplio conjunto de funciones para el manejo de cadenas.
 - Veremos las principales.

Delimitadores

Delimitadores de cadenas

- Una cadena está formada por cero o más caracteres encerrados entre comillas.
 - Pueden ser Dobles o Simples.
- Es obligatorio utilizar siempre el mismo tipo de comilla para una cadena.
- Se pueden entremezclar ambos, teniendo cuidado.

Delimitadores de cadenas

- Con comillas dobles: podemos incluir variables o caracteres especiales que **serán evaluados**.
- Esto no ocurre con las comillas simples.

Visualización de cadenas

Funciones de cadenas

- echo(cadena)
 - Muestra información por la salida estándar.
 - Sólo admite una única cadena como argumento.

Acceso al contenido

Acceso al contenido de una cadena

- Se puede acceder a cada uno de los caracteres que componen una cadena de la misma forma que lo hacemos con los elementos de un array.
 - Haciendo referencia a su posición en la cadena.
- `strlen (cadena)`: devuelve la longitud de la cadena.

Ejercicio

- Escribir una página en PHP que a partir de una cadena de caracteres muestre cada uno de los caracteres en una celda distinta de una tabla. Se debe indicar siempre en qué posición está el carácter en cuestión.

Carácter	Posición
S	0
a	1
l	2
u	3
d	4
o	5
s	6

```

<center>
    <h2> Función <i>strlen</i> </h2>
    <?php
        $cadena = "saludos";
        echo "<table border='1' cellpadding='2' cellspacing='2'>";
        echo "<tr bgcolor='yellow'><td>Carácter</td>";
        echo "<td>Posición</td></tr>";
        for($i=0; $i<strlen($cadena); $i++)
        {
            echo "<tr align='center'><td>". $cadena[$i];
            echo "</td><td>". $i . "</td></tr>";
        }
        echo "</table>";
    ?>
</center>

```

Función *strlen*

Carácter	Posición
s	0
a	1
l	2
u	3
d	4
o	5
s	6

Búsqueda en cadenas

Búsqueda en cadenas

- `stristr (cadena, buscar)`
 - Busca la cadena ‘buscar’ dentro de ‘cadena’
 - Devuelve la subcadena que va desde que se encuentra ‘buscar’ hasta el final de ‘cadena’.
 - Si no la encuentra devuelve una cadena vacía.
 - **Diferencia** entre mayúsculas y minúsculas.

Expresiones regulares

Expresiones regulares

- Las expresiones regulares son una potente herramienta que nos permite contrastar un texto con un **patrón de búsqueda**.
- Esta tarea resulta fundamental en algunos programas, y en otros puede facilitarnos increíblemente el trabajo.

Expresiones regulares

- Una expresión regular es una secuencia de caracteres que sirven de **patrón para comparar frente a un texto**.
- Se busca siempre comprobar si el texto contiene lo especificado en el patrón.
- Ejemplo:
 - Patrón: **in**
 - Coinciden:
 - intensidad
 - cinta
 - **interior**
 - Patrón: **[mp]adre**
 - Coinciden:
 - Mi **madre** se llama Luisa
 - Tu **padre** es jardinero

Expresiones regulares

- **Sintaxis y metacaracteres**

- **El punto**

- El punto representa **cualquier carácter**. Escribiendo un punto en un patrón querrás decir que ahí hay un carácter, cualquiera. Desde la A a la Z (en minúscula y mayúscula), del 0 al 9, o algún otro símbolo.

- **Ejemplos:**

- ca.a* coincide con *cana*, *cama*, *casa*, *caja*, etc...

- No coincide con *casta* ni *caa*

Expresiones regulares

- **Sintaxis y metacaracteres**
 - **Principio y fin de cadena**
 - Si queremos indicar al patrón qué es el principio de la cadena o qué es el final, debemos hacerlo con **^ para inicio y \$ para final**.
 - **Ejemplos:**

“**^olivas**” coincide con “**olivas verdes**”, pero no con “**quiero olivas**”

Expresiones regulares

- **Sintaxis y metacaracteres**
 - **Cuantificadores**
 - Indicar que cierto elemento del patrón va a repetirse un **número indeterminado de veces**.
 - Los cuantificadores son: **+** y ***** .
 - Usando **+** queremos decir que el elemento anterior aparece una o más veces.
 - Usando ***** queremos decir que el elemento anterior aparece cero o más veces.
- **Ejemplos:**
“*gafas+*” coincide con “*gafassss*” pero no con “*gafa*”

Expresiones regulares

- **Sintaxis y metacaracteres**
 - **Puede que esté (una vez) o puede que no:** ?
 - “coches?” coincide con “coche” y con “coches”
 - Para definir la **cantidad de veces que se repetirá el elemento**: { }
 - “abc{4}” coincide con “abcccc”, pero no con “abc” ni “abcc”,
 - “abc{1,3}” coincide con “abc”, “abcc”, “abccc”, pero no con “abcccc”
 - Si un parámetro queda vacío, significa “un número indeterminado”. Por ejemplo: “x{5,}” significa que la x ha de repetirse 5 veces, o más.

Expresiones regulares

- **Sintaxis y metacaracteres**

- **Rangos**

- Los corchetes [] permiten especificar el **rango de caracteres**.
 - Basta que aparezca *cualquiera de ellos*.

- **Ejemplos:**

- “c[ao]sa” coincide con “casa” y con “cosa”
 - “[a-f]” coincide con todos los caracteres alfabéticos de la “a” a la “f”
 - “[0-9][2-6][ANR]” coincide con “12A”, “35N”, “84R”,
 - Dentro de los corchetes,
 - el símbolo ^ ya no significa inicio, si no que es un negador.

Expresiones regulares

- **Sintaxis y metacaracteres**
 - **Alternancia**
 - Para alternar entre varias opciones, usaremos el símbolo |
 - Con este mecanismo haremos un disyuntor, que nos permitirá dar varias opciones.
 - Si una de ellas coincide, el patrón será cierto.
 - **Ejemplos:**
 - “*aleman(ia|es)*” coincide con “*alemania*” y con “*alemanes*”
 - “(*norte|sur|este|oeste*)” coincide con cualquiera de los puntos cardinales.

Expresiones regulares

- **Sintaxis y metacaracteres**
 - **Agrupadores**
 - Los paréntesis nos sirven para agrupar un subconjunto.
 - Es útil para definir la alternancia, pero agrupar un subpatrón nos permite trabajar con él como si fuera un único elemento.
 - **Ejemplos:**
 - “(abc)+” coincide con “abc”, “abcabc”, “abcababcabc”, etc
 - “ca(sca)?da” coincide con “cascada” y con “cada”

Expresiones regulares

- **Sintaxis y metacaracteres**

- **Escapar caracteres**

- Si queremos que en el patrón hubiese un punto, o un símbolo asterisco, sin que se interprete como metacarácter, tendremos que “escaparlo”.
 - Esto se hace poniendo una barra invertida justo antes: \. o *
 - Esto puede hacerse con cualquier carácter que quieras introducir de **forma literal**, y no interpretada.

Búsqueda con expresiones regulares

Búsqueda con expresiones regulares

- `preg_match(patrón, cadena)`
 - Comprueba si una cadena cumple con un patrón dado.
 - El patrón debe ser una expresión regular.
 - El patrón será una cadena que empezará y terminará con barra (/)

Ejemplos

- Comprobar si una cadena contiene alguna “r”

```
<?php  
    $cad="riesgo";  
    if (preg_match("/ r /", $cad)) echo "SI";  
    else echo "NO";  
?>
```

- Comprobar si una cadena contiene algún número

```
<?php  
    $cad="moto";  
    if (preg_match("/ [1-9] /", $cad)) echo "SI";  
    else echo "NO";  
?>
```

Ejemplos

- Comprobar si una cadena tiene dos números

```
<?php  
    $cad="riesgo33";  
    if (preg_match("/[0-9]{2}/", $cad)) echo "SI";  
    else echo "NO";  
?>
```

- Comprobar si una cadena empieza por dos números

```
<?php  
    $cad="riesgo33";  
    if (preg_match("/^([0-9]{2})/", $cad)) echo "SI";  
    else echo "NO";  
?>
```

Ejemplos

- Comprobar si una cadena termina por S

```
<?php  
    $cad="riesgo33S";  
        if (preg_match("/\$/", $cad)) echo "SI";  
        else echo "NO";  
?>
```

- Comprobar que una cadena tenga un número después una letra minúscula y después un número

```
<?php  
    $cad="riesgo3T3S";  
        if (preg_match("/[0-9][a-z][0-9]/", $cad)) echo "SI";  
        else echo "NO";  
?>
```

Ejemplos

- Comprobar que una cadena tenga un número, después una sola letra mayúscula o minúscula y después otro número

```
<?php
$cad="riesgo3T3S";
if (preg_match("/[0-9]([a-z]|[A-Z])[0-9]/", $cad)) echo "SI";
else echo "NO";
?>
```

- Comprobar que una cadena tenga un número después dos o más letras minúsculas y después la A

```
<?php
$cad="e3seA3S";
if (preg_match("/[0-9][a-z]{2,}A/", $cad)) echo "SI";
else echo "NO";
?>
```

Comparación de cadenas

Comparación de cadenas

- `strcmp (cad1, cad2):`
 - Compara ambas cadenas y devuelve
 - Valor < cero, si cad1 es va antes que cad2.
 - Valor > cero, si cad1 es va después que cad2.
 - Cero, si ambas son iguales.
 - **Distingue** entre mayúsculas y minúsculas.
- `strcasecmp (cad1, cad2):`
 - Igual que la anterior, pero **NO** diferencia entre mayúsculas y minúsculas.

```
<?php
```

```
$cad1 = "Saludos";  
$cad2 = "saludos";  
echo "<table border='1' cellpadding='2' cellspacing='2'>";  
echo "<tr align='center'><td bgcolor = 'pink'>cadena 1</td>";  
echo "<td>$cad1</td></tr>";  
echo "<tr align='center'><td bgcolor = 'pink'>cadena 2</td>";  
echo "<td>$cad2</td></tr>";  
echo "<tr align='center'><td bgcolor = 'pink'>strcmp(cad1, cad2) ">  
echo "</td><td>".strcmp($cad1, $cad2)."</td></tr>";  
echo "<tr align='center'><td bgcolor = 'pink'>strcasecmp(cad1, cad2) ">  
echo "</td><td>".strcasecmp($cad1, $cad2)."</td></tr>";  
echo "</table>";
```

```
?>
```

Funcion *strcmp* y *strcasecmp*

cadena 1	Saludos
cadena 2	saludos
strcmp(cad1, cad2)	-1
strcasecmp(cad1, cad2)	0

Comparación de cadenas

- `strncmp (cad1, cad2, num):`
 - Igual que `strcmp()` pero sólo compara los ‘num’ primeros caracteres de cada cadena.

Operar con subcadenas

Operar con subcadenas

- `substr (cad, inicio [,tam])`
 - Devuelve la subcadena que va desde ‘inicio’ hasta el fin, o si se indica, el número de caracteres indicados.
- `substr_replace(cad1, cad2, inicio [,tam])`
 - Devuelve una cadena que es el resultado de sustituir dentro de cad1 el trozo que va desde ‘inicio’ hasta el final (o el número de caracteres) por cad2
 - La cadena original no sufre modificación.

Operar con subcadenas

- `str_replace(cadbus, cadree, cadena)`
 - Devuelve una cadena en la que todas las apariciones de ‘cadbus’ se cambian por ‘cadree’ en cadena.
 - La cadena original no sufre cambios.
- `strtr (cadena, cadbus, cadree)`
 - Igual que `str_replace`, pero se cambian cada uno de los caracteres de la cadena buscada, por su correspondiente en la cadena de sustitución.

```
<?php
```

```
$cadena = "abcdefghijkl";  
$cadB = "aei";  
$cadS = "AEI";  
echo "<table border='1' cellpadding='2' cellspacing='2'>";  
echo "<tr align='center'><td bgcolor = 'pink'>cadena</td>";  
echo "<td>$cadena</td></tr>";  
echo "<tr align='center'><td bgcolor = 'pink'>Patrón</td>";  
echo "<td>$cadB</td></tr>";  
echo "<tr align='center'><td bgcolor = 'pink'>Sustitución</td>";  
echo "<td>$cadS</td></tr>";  
echo "<tr align='center'><td bgcolor = 'pink'>";  
echo "strtr(cadena,patrón, sustitucion)";  
echo "</td><td>".strtr($cadena, $cadB, $cadS). "</td></tr>";  
echo "</table>";
```

```
?>
```

Funcion *strtr*

cadena	abcdefghijkl
Patrón	aei
Sustitución	AEI
strtr(cadena,patrón, sustitucion)	AbcdEfgHIjkl

Operar con subcadenas

- `substr_count(cadena, buscar)`
- Devuelve el número de veces que aparece ‘buscar’ en ‘cadena’

Modificación del contenido

Limpieza de cadenas

Limpieza de cadenas

- trim (cadena)
 - Devuelve la cadena pero elimina los espacios en blanco del principio y del final de dicha cadena.
- rtrim (cadena)
 - Devuelve la cadena sin los espacios en blanco que haya al final de la cadena. (right)
- ltrim(cadena)
 - Devuelve la cadena pero elimina los espacios en blanco al principio de la cadena. (left)

Relleno de cadenas

- **str_pad** (`cadena`, `long`, `car`, `modo`)
 - Rellena una cadena con un carácter de relleno (por defecto espacio en blanco) hasta que la cadena tenga la longitud deseada.
 - Opcionalmente se puede indicar el **modo** de relleno:
 - `STR_PAD_RIGHT`: relleno por la derecha (defecto)
 - `STR_PAD_LEFT`: relleno por la izquierda.
 - `STR_PAD_BOTH`: relleno por ambos lados, intenta colocar los mismos caracteres a derecha e izquierda.

Modificación del contenido

Conversión entre mayúsculas y minúsculas

Conversión Mayúsculas Minúsculas

- `strtolower(cadena)`
 - Convierte una cadena de caracteres a minúsculas.
- `strtoupper(cadena)`
 - Convierte una cadena de caracteres a mayúsculas.
- `ucfirst(cadena)`
 - Convierte a mayúsculas el primer carácter de una cadena.
- `ucwords(cadena)`
 - Convierte a mayúsculas el primer carácter de cada palabra de la cadena.

Otras funciones con cadenas

Otras funciones

- `strrev(cadena)`
 - Devuelve la cadena invertida.
- `str_repeat(cadena, veces)`
 - Devuelve una cadena con la cadena que se le pasa repetida tantas veces como se pase en el 2º parámetro.

Manejo de Cadenas

Programando con PHP

Programando con PHP

FECHAS Y HORAS

Fechas y horas

Introducción

Introducción

- En la Web se hace necesario llevar un control de la fecha y la hora.
- PHP ofrece un gran número de funciones de control de fechas y horas.
- Veremos algunas de ellas...

Introducción

- En casi todos los sistemas, hay una fecha común a partir de la cual se empieza a contar el tiempo.
- La fecha elegida es 01/01/1970 a las 00:00
- En PHP, todas las funciones de fecha/hora hacen referencia a esta fecha.

Introducción

- Existe un tipo de dato (**timestamp** o **marca de tiempo**) que hace referencia a esta fecha.
- La marca de tiempo almacena el número de segundos transcurridos desde 01/01/1970 hasta una fecha dada.

Introducción

- Así, el día 02/01/1970 a las 00:00, realmente se almacena en memoria con su **timestamp** asociado: **86400** que son los segundos que han pasado desde las 00:00 del 01/01/1970.
- Las 00:00:01 del día 02/01/1970 tiene como timestamp asociado el número: **86401**.
- Y así sucesivamente...

Introducción

- El uso de timestamp hace que la comparación de fechas y horas sea muy **exacta**.
- Esto hace que sea necesario tener funciones que nos transformen los timestamp en fecha legibles para los humanos.
- PHP incluye multitud de funciones que persiguen este objetivo.

Funciones de fecha y hora

Resumen de funciones

Función	Descripción
time()	Devuelve la hora actual
checkdate()	Valida una fecha en formato gregoriano
date()	Da formato a la fecha y hora locales
getdate()	Obtiene información sobre la fecha y hora locales
gettimeofday()	Obtiene la hora actual
gmdate()	Formatea la fecha y hora a formato GMT
gmmktime()	Obtienen la marca de tiempo UNIX de una fecha con formato GMT
microtime()	Obtiene la hora actual en microsegundos

Funciones de fecha y hora

● time()

- Devuelve la marca de tiempo actual.

```
<?php  
  
    $marca_hoy = time();  
  
    echo $marca_hoy;  
?>
```

1422954668

Funciones de fecha y hora

○ checkdate(mes, dia, anio)

- Verifica si la fecha que se le pasa como parámetro es una fecha correcta.
- Útil para formularios.
- Comprueba que el número de día sea correcto teniendo en cuenta el mes y el año.
- Devuelve *true* o *false*.

Funciones de fecha y hora

○ Date (formato, [timestamp])

- Permite darle un formato específico a una cadena que tendrá una fecha y una hora.
- Acepta una cadena de formato y un parámetro *timestamp*.
- Si no lo ponemos, tomará la hora actual.
- Normalmente se omite, y se introduce sólo el formato:

Caracteres de formato para día

Carácter	Descripción	Ejemplo
d	Día del mes, 2 dígitos con ceros iniciales	01 a 31
D	Representación textual de un día, 3 letras	Mon hasta Sun
j	Día del mes sin ceros iniciales	1 a 31
l ('L' minúscula)	Una representación textual completa	Sunday hasta Saturday
N	Representación numérica	1 (para lunes) hasta 7 (para domingo)
w	Representación numérica del día de la semana	0 (domingo) hasta 6 (sábado)
z	El día del año (comenzando por 0)	0 hasta 365
W	Número de la semana del año	Ejemplo: 42 (la 42 ^a semana del año)

Ejemplo

- Mostrar el día de la semana correspondiente al día actual

```
<?php  
    /*no paso timestamp  
para tomar fecha actual*/  
    $dia=date('l');  
    echo "Hoy es $dia";  
?>
```

Hoy es Tuesday

Caracteres de formato para mes y año

Carácter	Descripción	Ejemplo
F	Una representación textual completa de un mes	January hasta December
m	Representación numérica de un mes 01 hasta 12	
M	Una representación textual corta de un mes	Jan hasta Dec
n	Representación numérica de un mes, sin 0	1 hasta 12
t	Número de días del mes dado	28 hasta 31
L	Si es un año bisiesto	1 si es bisiesto, 0 si no.
Y	Una representación numérica completa de un año, 4 dígitos	Ejemplos: 1999 o 2003
y	Una representación de dos dígitos de un año	Ejemplos: 99 o 03

Ejemplo

- Mostrar si el año actual es o no bisiesto

```
<?php
    /*no paso timestamp
    para tomar fecha actual*/
    $bisi=date('L');
    if($bisi==1)
    {
        echo "Este año es bisiesto";
    }
    else
    {
        echo "Este año no es bisiesto";
    }

?>
```

Caracteres de formato para hora

Carácter	Descripción	Ejemplo
a	am o pm en minúsculas	am o pm
A	am o pm en mayúsculas	AM o PM
B	Hora Internet	000 hasta 999
g	Formato de 12 horas de una hora sin ceros iniciales	1 hasta 12
G	Formato de 24 horas de una hora sin ceros iniciales	0 hasta 23
h	Formato de 12 horas de una hora con ceros iniciales	01 hasta 12
H	Formato de 24 horas de una hora con ceros iniciales	00 hasta 23
i	Minutos, con ceros iniciales	00 hasta 59
s	Segundos, con ceros iniciales	00 hasta 59
u	Microsegundos (añadido en PHP 5.2.2)	Ejemplo: 654321

Caracteres de formato para zona horaria

Carácter	Descripción	Ejemplo
e	Identificador de zona horaria	Ejemplos: UTC, GMT, Atlantic/Azores
I (i mayúscula)	Si la fecha está en horario de verano o no	1 si está en horario de verano, 0 si no.
O	Diferencia de la hora de Greenwich (GMT) en horas	Ejemplo: +0200
P	Diferencia con la hora de Greenwich (GMT) con dos puntos entre horas y minutos (añadido en PHP 5.1.3)	Ejemplo: +02:00
T	Abreviatura de la zona horaria	Ejemplos: EST, MDT ...

Ejemplo

- Se pueden hacer combinaciones de los formatos para obtener el formato que nos interesa:
 - Mostrar la fecha de hoy en formato dd-mm-yyyy

```
<?php  
    $fecha = date ('d-m-Y') ;  
    echo "<br>Hoy es $fecha" ;  
?>
```

Hoy es 03-02-2015

Funciones de fecha y hora

◎ getdate(timestamp)

- Devuelve un array asociativo que contiene información sobre la fecha y hora asociadas a la marca de tiempo pasada.
- En caso de no pasarle nada, toma la marca de tiempo actual.
- El array asociativo devuelto es:

Clave	Contenido
Seconds	Identifica los segundos
Minutes	Identifica los minutos
Hours	Identifica las horas
Mday	Día del mes
Wday	Día de la semana (en número)
Mon	mes en número
Year	Año
Yday	Número de día del año
Weekday	Día de la semana (nombre)
Month	Mes (en nombre)

Funciones de fecha y hora

- mktime (hora, min, seg, mes, dia, anio)
 - Devuelve la marca de tiempo (segundos pasados desde las 00:00:00 del día 01/01/1970) correspondiente a los valores pasados.
 - Muy útil para realizar cálculos matemáticos con fechas.
 - También útil para validaciones de fechas.

Ejemplo

- Obtener el día de la semana correspondiente a la fecha 17/05/2004

```
<?php  
    $marca = mktime(0, 0, 0, 5, 17, 2004);  
  
    $dia_semana = date('l', $marca);  
    echo "El dia 17/05/2004 fue $dia_semana";  
?>
```

El dia 17/05/2004 fue Monday

Funciones de fecha y hora

● strtotime (fecha)

- Devuelve la marca de tiempo correspondiente a la fecha recibida.
- La fecha se recibe en una cadena de texto
- La cadena de texto debe representar una fecha en inglés:
 - 4 números para el año
 - 2 números para el mes
 - 2 números para el día
 - Separados por guiones
- 2015-02-21

Ejemplo

```
<?php  
    $time = strtotime('2015-12-21');  
    $dia = date('d', $time);  
    $mes = date('m', $time);  
    $anio = date('Y', $time);  
    echo "<br>Hoy es $dia de $mes de $anio";  
?>
```

Hoy es 21 de 12 de 2015

Programando con PHP

FECHAS Y HORAS



Programando con PHP

FUNCIONES

Introducción

Funciones

Introducción

- Sección de código separada a la que se le ha dado un nombre.
- Puede ser llamada tantas veces como se quiera.
- Dividen tareas para hacer más comprensible el script PHP
- Pueden recibir / devolver valores

Introducción

- Sintaxis:

```
Function nombreFuncion (parametros)
{
    Sentencias
}
```

Introducción

- Para llamar a una función sólo hay que escribir una sentencia que contenga el nombre de la función, seguida de los parámetros.
- La ejecución salta a la línea en la que empieza la función.

Ejemplo

```
<?php  
function cuentaAtras()  
{  
    for ($i=10; $i>0; $i--)  
    {  
        echo $i."...<br>";  
    }  
    echo "¡ Boooooommm  
}  
?  
?>
```

```
<td bgcolor = '#FFBBAA'>  
<?php  
    cuentaAtras();  
?>  
</td>  
<td bgcolor = 'pink'>  
<?php  
    cuentaAtras();  
?>
```

Ejemplo

10...

9...

8...

7...

6...

5...

4...

3...

2...

1...

| Boooooommmmm !

10...

9...

8...

7...

6...

5...

4...

3...

2...

1...

| Boooooommmmm !

Introducción

- En la mayoría de las ocasiones, necesitaremos parámetros.
- Ejemplo...

Ejemplo

```
<?php
    function cuentaAtras($inicio)
    {
        for ($i=$inicio; $i>0; $i--)
        {
            echo $i."...<br>";
        }
        echo "<td bgcolor = '#FFBBAA'>
              ?>
              cuentaAtras(8);
              ?>
              </td>
              <td bgcolor = 'pink'>
              ?>
              cuentaAtras(15);
              ?>
```

Ejemplo

8...

7...

6...

5...

4...

3...

2...

1...

| Boooooommmmm !

15...

14...

13...

12...

11...

10...

9...

8...

7...

6...

5...

4...

3...

2...

1...

| Boooooommmmm !

Paso de parámetros

Funciones

Paso de parámetros

- PHP permite pasar los parámetros:
 - Por valor
 - Por referencia
 - Por defecto

Paso de parámetros por valor

- Es la opción por defecto en PHP
- La función recibe una copia del valor de la variable.
- No hay problema en modificarla.

Ejercicio

- Crear una función PHP que reciba dos números y muestre por pantalla la suma de ellos
- Crear una función PHP que reciba un número y muestre por pantalla la tabla de multiplicar de ese número

Paso de parámetros por defecto

- Son la forma en que PHP implementa los parámetros opcionales.
- Toman el valor predefinido, cuando al llamar la función, no se les pasa un argumento.

Paso de parámetros por defecto

- Para definirlos hay que escribir, en la cabecera de la función:
 - Nombre del parámetro
 - Operador de asignación (=)
 - Valor por defecto
- Los parámetros pasados por defecto deben ir **SIEMPRE** los últimos.

Ejemplo

```
<?php  
function cuentaAtras($inicio, $fin, $mensaje="¡Booomm!")  
{  
    for ($i=$inicio; $i>$fin; $i--)  
    {  
        echo $i."...<br>";  
    }  
    echo $mensaje;      <td bgcolor = '#FFBBAA'>  
}                                <?php  
?>                                cuentaAtras(8,4);  
?>  
</td>  
<td bgcolor = 'pink'>  
<?php  
cuentaAtras(12,7,"¡YAAAAA!");  
?>
```

Ejemplo

8...

7...

6...

5...

|Boooomm!

12...

11...

10...

9...

8...

|YAAAAA!

■■■ Devolución de variables

Funciones

Devolución de valores

- Se pueden devolver cualquier tipo de valor.
- SÓLO se puede devolver un valor
- Para devolver varios valores, devolveremos un array.
- Utilizaremos la palabra reservada *return*
- Cuando se encuentra un *return* la función termina.

Ejemplo

```
<?php
```

```
FUNCTION sumar ($num1, $num2)
{
    $resultado = $num1 + $num2;

    return $resultado;
}
```

```
?>
```

```
<?php
```

```
$n1 = 25;
$n2 = 36;
```

```
$resultado_suma = sumar($n1, $n2);
```

```
echo "$n1 + $n2 = $resultado_suma";
```

A screenshot of a web browser window. The address bar shows "localhost/suma.php". The main content area displays the output of the PHP script: $25 + 36 = 61$.

```
localhost/suma.php
← → C localhost/suma.php
25 + 36 = 61
```

Incluir funciones

Funciones

Incluir Funciones

- Como hemos visto, un función sólo está disponible en el archivo en el que se ha creado.
- Esto no es eficiente ya que lo normal es que se utilicen en muchos ficheros diferentes.
- Lo normal es crear un único documento donde estén almacenadas todas las funciones que se vayan a utilizar en la Web
- Con la sentencia **include** se indicará a PHP dónde buscar las funciones.

Documento “funciones.php”

```
<?php
    function SumaNumeros ($n1, $n2)
    {
        $suma = $n1+$n2;

        echo "la suma de $n1 + $n2 es $suma<br><br>";
    }
    function TablaMultiplicar ($a)
    {
        $resul=0;
        for ($i=0; $i<=10; $i++)
        {
            $resul = $a * $i;
            echo "$a x $i = $resul<br>";
        }
    }
?>
```

Uso del documento

```
<?php  
  
include ("funciones.php");  
  
SumaNumeros(4, 15);  
  
echo "<br><br>";  
  
SumaNumeros(8, 3);  
  
echo "<br><br>";  
  
TablaMultiplicar(4);  
  
echo "<br><br>";  
  
TablaMultiplicar(15);  
  
?>
```

la suma de 4 + 15 es 19

la suma de 8 + 3 es 11

$4 \times 0 = 0$

$4 \times 1 = 4$

$4 \times 2 = 8$

$4 \times 3 = 12$

$4 \times 4 = 16$

$4 \times 5 = 20$

$4 \times 6 = 24$

$4 \times 7 = 28$

$4 \times 8 = 32$

$4 \times 9 = 36$



Programando con PHP

FUNCIONES

Bases de Datos

Bases de Datos

» Definiciones

Definiciones

- ▶ Una **Base de Datos** es una colección de datos que se encuentran almacenados juntos, organizados y relacionados entre sí.
- ▶ Un **Sistema Gestor de Bases de Datos** es el software que nos ayuda a comunicarnos con una Base de Datos

Definiciones

- ▶ Existe muchos tipos de SGBD, el más utilizado es el SGBD relacionales.
- ▶ Las Bases de Datos relacionales están compuestas por:
 - Tablas
 - Filas
 - Columnas
 - Relaciones entre las tablas

Bases de Datos Relacionales

» Elementos

Tablas

- ▶ Una Base de Datos **relacional** tiene como elemento principal las tablas.
- ▶ Las tablas estarán compuestas por una serie de **columnas** (atributos) y una serie de **filas** (datos).
- ▶ Cada tabla representa a un conjunto de datos de la realidad.

Tablas

- ▶ Cada **columna** de una tabla debe tener:
 - Un nombre
 - Un tipo de dato
 - Una longitud
- ▶ Se debe de poder **diferenciar** de forma univoca a cada elemento del conjunto.
- ▶ Cada tabla debe tener una **clave principal**
 - Una columna de la tabla que permite diferenciar a todos los elementos del conjunto sin lugar a error.

Tablas

- ▶ Para que un atributo pueda ser clave principal debe cumplir que:
 - No se pueda **repetir** el mismo valor en elementos distintos.
 - No pueda **cambiar** a lo largo del tiempo
 - No puede estar **vacío**
- ▶ La clave principal de una tabla no tiene por qué ser un único atributo, puede estar compuesta por **varios atributos juntos**.

Ejemplos

PERSONA

Nombre	DNI	Teléfono

LIBRO

ISBN	Título	Autor	Año publicación

FECHA

Día semana	Día	Mes	Año

MASCOTA

Nombre	Edad	Peso	Raza	Color

VEHICULO

Matrícula	Color	Marca	Modelo	Precio

Relaciones

- ▶ Una Base de Datos es una colección de datos que se encuentran almacenados juntos, organizados y relacionados entre sí.
- ▶ Algunas de las tablas que componen una Base de Datos pueden tener **relación** entre sí

PERSONA

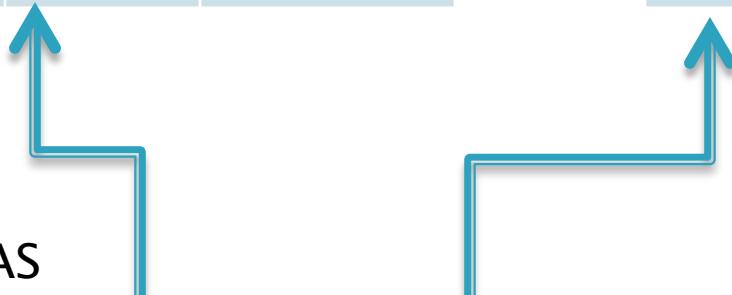
Nombre	DNI	Teléfono

LIBRO

ISBN	Título	Autor	Año publicación

VENTAS

Persona	Libro	Unidades	Fecha



Relaciones

- ▶ Estas relaciones se traducen en:
 - Para que un elemento pueda existir en una tabla que hace referencia a otra, antes deberá existir en la tabla referenciada.
- ▶ En nuestro ejemplo
 - Para que una persona pueda comprar un libro, antes debemos tener registrada a la persona y al libro

Creando Bases de Datos

» El lenguaje SQL

El lenguaje SQL

- ▶ **SQL** (Structured Query Language) es un lenguaje de consulta de Bases de Datos que nos permite realizar cualquier operación que queramos.
- ▶ Vamos a ver como:
 - Crear tablas
 - Insertar datos en las tablas
 - Modificar datos de las tablas
 - Eliminar datos de las tablas
 - Consultar información de las tablas

EL lenguaje SQL

- ▶ Las operaciones de SQL reciben el nombre de **sentencias**.
- ▶ Están formadas por diferentes partes que denominamos **cláusulas**.

Sentencia ◀ [SELECT codigo_producto, nombre_producto, tipo ————— Cláusula
 | FROM productos ————— Cláusula
 | WHERE precio > 1000; ————— Cláusula

El lenguaje SQL y MySQL

» Creando una Base de Datos

MySQL

- ▶ MySQL es un SGBD relacional, libre y de código abierto.
- ▶ XAMPP viene con él instalado.
- ▶ Para acceder a panel de control de MySQL, escribiremos en el navegador:

localhost/phpmyadmin

MySQL

► Accederemos a una página parecida esta:

The screenshot shows the phpMyAdmin interface running on a local host. The top navigation bar includes links for Bases de datos, SQL, Estado actual, Usuarios, Exportar, Importar, Configuración, Replicación, Variables, Juegos de caracteres, and Motores. On the left, a sidebar lists databases: cdcoll, data, information_schema, mysql, performance_schema, phpmyadmin, test, and webauth. The main content area is divided into several sections:

- Configuraciones generales:** Shows the character set for the connection as utf8_general_ci.
- Configuraciones de apariencia:** Shows the language as Español - Spanish, theme as pmahomme, and font size as 82%.
- Servidor de base de datos:** Displays server details: Servidor: 127.0.0.1 via TCP/IP, MySQL Community Server (GPL), Version: 5.6.11, Protocol: 10, User: root@localhost, and Character Set: UTF-8 Unicode (utf8).
- Servidor web:** Displays web server details: Apache/2.4.4 (Win32) OpenSSL/1.0.1e PHP/5.5.3, MySQL Version: libmysql - mysqlnd 5.0.11-dev - 20120503 - \$Id: 40933630dede551dfaca71298a83fad8d03d62d4 \$, and PHP Extension: mysqli.
- phpMyAdmin:** A list of links including Acerca de esta versión, Documentación, Wiki, Página oficial de phpMyAdmin, Contribuir, Obtener soporte, and Lista de cambios.

A message at the bottom left states: "Una versión más reciente de phpMyAdmin está disponible y le recomendamos que la obtenga. La versión más reciente es 4.1.6, y existe desde el 2014-01-26."

A red warning box at the bottom right contains the following text:
! Su archivo de configuración contiene parámetros (root sin contraseña) que corresponden a la cuenta privilegiada predeterminada de MySQL. Su servidor de MySQL está usando estos valores, lo que constituye una vulnerabilidad. Se le recomienda corregir esta brecha de seguridad. Por ejemplo, desde la página de inicio de phpMyAdmin seleccione Privilegios y agregue la contraseña a root@localhost. Deberá escribir la misma contraseña en config.inc.php de phpMyAdmin.

MySQL

- ▶ Lo primero que debemos hacer es **crear una base de datos** donde estarán nuestras tablas.

- ▶ Para ello:

- Pulsamos en el botón Bases de Datos



- Elegimos un nombre para la Base de Datos y le damos a Crear



- En ese momento aparecerá la nueva Base de Datos en el árbol de la izquierda

El lenguaje SQL y MySQL

» Creando tablas

Creación de Tablas. Pasos a seguir

1. Decidir el **nombre** de la tabla
2. Nombre de cada uno de las **columnas**
3. A cada una de las columnas asignar un **tipo de datos**
 - ✓ También podremos dar definiciones por defecto y restricciones de columna.
4. Sólo nos quedará dar las **relaciones** de la tabla.

Creación de tablas

- ▶ Crear tablas en desde phpMyAdmin es también muy sencillo.
- ▶ Una vez seleccionada la Base de Datos con la que vamos a trabajar nos aparece un formulario donde podemos elegir el nombre de la tabla y el número de filas que va a tener.

The screenshot shows a web-based form titled "Crear tabla" (Create Table). At the top left is a small icon of a table with a yellow starburst. To its right is the title "Crear tabla". Below the title are two input fields: "Nombre:" (Name:) followed by a text input box, and "Número de columnas:" (Number of columns:) followed by another text input box. At the bottom right of the form is a rounded rectangular button labeled "Continuar" (Continue).

Nombre:

Número de columnas:

Continuar

Columnas. Tipos de datos

- ▶ Una vez creada la tabla, nos aparece un formulario con una fila por cada una de las columnas de la tabla

Nombre	Tipo 	Longitud/Valores 	Predeterminado 
<input type="text"/>	INT 	<input type="text"/>	Ninguno 

- ▶ Los **tipos de datos** más utilizados son:
 - varchar o text → cadenas de texto
 - int → números sin decimales
 - Decimal → números con decimales
 - date o timestamp → fechas
 - Set → Conjunto de valores exactos
 - Serial → Campo autonumérico de 1 en adelante

Columnas. Tipos de datos

- ▶ En el campo Longitud/Valores se debe introducir una de estas 3 cosas
 - **Cadenas y enteros**: longitud máxima
 - **Decimales**: Número de dígitos totales, Número de decimales (dig,dec)
 - **Set**: Los valores que podrá contener la columna
 - **Serial** y **Date**: no llevan valor en este campo

Restricciones de columna

- ▶ Una vez elegidos las columnas y asignados los tipos de datos, hay que decidir si las columnas tienen alguna **restricción**.
- ▶ Las restricciones pueden ser del tipo:
 - Debe ser **único** en toda la tabla o puede repetirse.
 - Puede quedar **vacío** o debe estar siempre lleno.
 - Si es o no **clave** principal

Restricciones de columna

Restricción	descripción
NULO	En la columna no puede haber valores nulos
UNIQUE	En toda la columna no se pueden repetir valores
PRIMARY	La columna es clave primaria: no puede tener valores nulos ni repetidos.

Creación de tablas: Ejercicios

- ▶ La Base de Datos “Productora” contiene la siguientes tablas:

Estrella (nombre, dirección, sexo, f_nacimiento)

Estudio (nombre, dirección)

Película (titulo, año, duración, nombre_estudio)

Protagonistas (título película, año pel, nomb estr)

PRODUCTORA: Estrella y estudio

ESTRELLA

Nombre	Tipo 	Longitud/Valores 	Predeterminado 	Cotejamiento	Atributos	Nulo	Índice
nombre	VARCHAR 	20	Ninguno 			<input checked="" type="checkbox"/>	PRIMARY 
direccion	VARCHAR 	50	Ninguno 			<input checked="" type="checkbox"/>	--- 
sexo	SERIAL 	h, m	Ninguno 			<input checked="" type="checkbox"/>	--- 
f_nacimiento	DATE 		Ninguno 			<input checked="" type="checkbox"/>	--- 

ESTUDIO

Nombre	Tipo 	Longitud/Valores 	Predeterminado 	Cotejamiento	Atributos	Nulo	Índice
nombre	VARCHAR 	20	Ninguno 			<input checked="" type="checkbox"/>	PRIMARY 
direccion	VARCHAR 	50	Ninguno 			<input checked="" type="checkbox"/>	---

PRODUCTORA: películas y protagonistas

PELÍCULAS

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	Índice
titulo	VARCHAR	50	Ninguno			<input type="checkbox"/>	PRIMARY
anio	INT	4	Ninguno			<input type="checkbox"/>	PRIMARY
duracion	INT	3	Ninguno			<input type="checkbox"/>	---
nombre_estudio	VARCHAR	20	Ninguno			<input type="checkbox"/>	---

PROTAGONISTAS

Nombre	Tipo	Longitud/Valores	Predeterminado	Cotejamiento	Atributos	Nulo	Índice
titulo_pelicula	VARCHAR	50	Ninguno			<input type="checkbox"/>	PRIMARY
anio_peli	INT	4	Ninguno			<input type="checkbox"/>	PRIMARY
nomb_estr	VARCHAR	20	Ninguno			<input type="checkbox"/>	PRIMARY

Relaciones de las tablas

- ▶ Una vez creadas todas las tablas necesarias en nuestra Base de Datos, habrá que relacionar unas con otras.
- ▶ Para ello seleccionamos la tabla que va a hacer referencia a otras y nos vamos a **SU** sección Estructura

<input type="checkbox"/> estudio	 Examinar	 Estructura	
<input checked="" type="checkbox"/> pelicula	 Examinar	 Estructura	
<input type="checkbox"/> protagonistas	 Examinar	 Estructura	

Relaciones de las tablas

- ▶ Dentro de la tabla, seleccionamos Vista de relaciones

Examinar Estructura SQL

#	Nombre	Tipo	Cotejamiento
1	<u>titulo</u>	varchar(50)	latin1_swedish
2	<u>anio</u>	int(4)	
3	<u>duracion</u>	int(3)	
4	<u>nombre_estudio</u>	varchar(20)	latin1_swedish

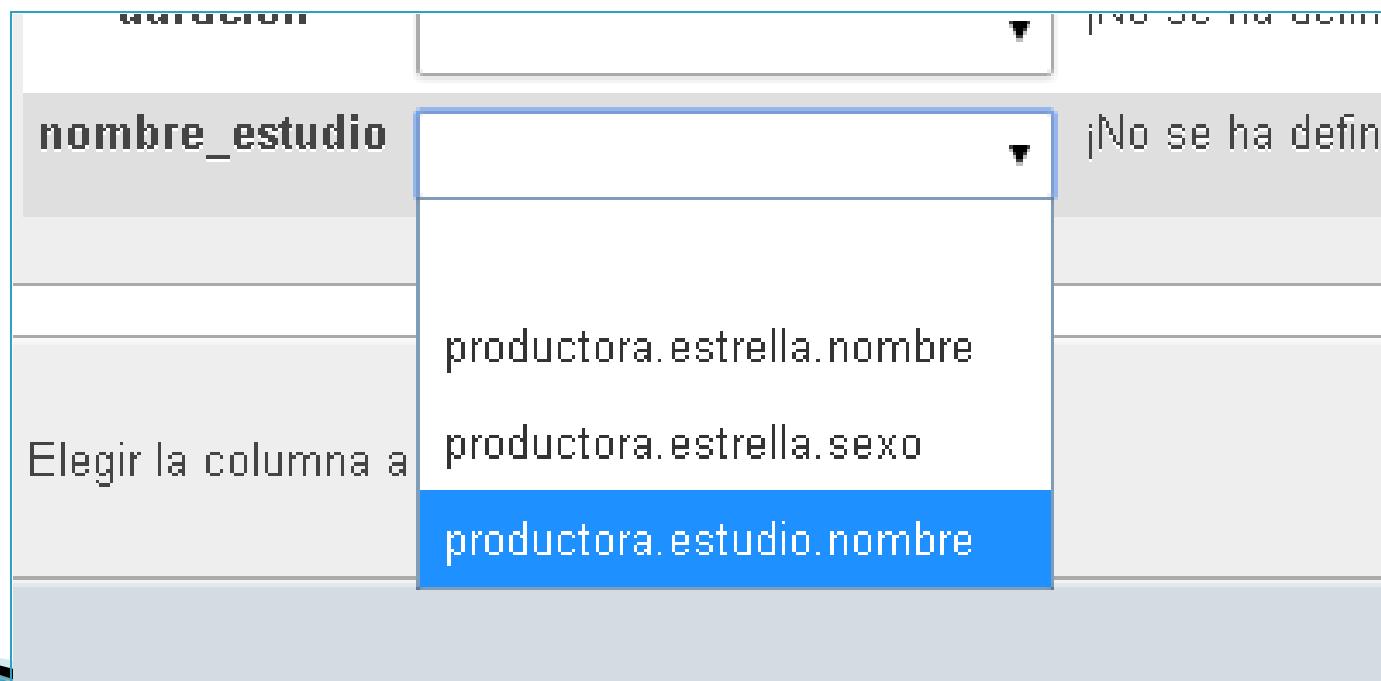
↑ Marcar todos Para los elementos que e
- Eliminar Primary Unique In

Vista de impresión Vista de relaciones Plan
Hacer seguimiento a la tabla Mover columnas

Vista de relaciones

Relaciones de las tablas

- ▶ Una vez aquí sólo hay que elegir dentro de la lista desplegable, a qué campo va a hacer referencia cada una de las columnas



Ejercicios



Creación de tablas: Ejercicios

Persona (nombre, calle, ciudad)

Trabajador (nombre, nombre_empresa, salario)

Empresa (nombre empresa, ciudad)

Director_empresa (nombre empresa, nombre director)

NOTA: El salario por defecto será de 1000€

Crear base de datos

empresa

Cotejamiento

Crear



Crear tabla

Nombre: persona

Número de columnas: 3

Continuar



Nombre	Tipo 	Longitud/Valores 	Predeterminado 	Cotejamiento	Atributos	Nulo	Índice
nombre	VARCHAR 	15	Ninguno 			<input type="checkbox"/>	PRIMARY 
calle	VARCHAR 	20	Ninguno 			<input type="checkbox"/>	---
ciudad	VARCHAR 	20	Ninguno 			<input type="checkbox"/>	---

 Crear tabla

Nombre: Número de columnas:



Nombre	Tipo 	Longitud/Valores 	Predeterminado 	Cotejamiento	Atributos	Nulo	Índice
nombre	VARCHAR 	<input type="text" value="15"/>	Ninguno 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	PRIMARY 
compañía	VARCHAR 	<input type="text" value="20"/>	Ninguno 	<input type="text"/>	<input type="text"/>	<input type="checkbox"/>	-- 
salario	DECIMAL 	<input type="text" value="6,2"/>	Personalizado: 	<input type="text"/> 1000	<input type="text"/>	<input type="checkbox"/>	-- 

 Crear tabla

Nombre: Número de columnas:



Estructura 

Nombre	Tipo 	Longitud/Valores 	Predeterminado 	Cotejamiento	Atributos	Nulo	Índice
compañía	VARCHAR 	<input type="text" value="15"/>	<input type="text" value="Ninguno"/>			<input type="checkbox"/>	PRIMARY 
ciudad	VARCHAR 	<input type="text" value="15"/>	<input type="text" value="Ninguno"/>			<input type="checkbox"/>	-- 

 Crear tabla

Nombre: director_empresa Número de columnas: 2



Nombre	Tipo 	Longitud/Valores 	Predeterminado 	Cotejamiento	Atributos	Nulo	Índice
compañía	VARCHAR 	15	Ninguno 			<input type="checkbox"/>	PRIMARY 
nombre_director	VARCHAR 	15	Ninguno 			<input type="checkbox"/>	-- 

Creación de las relaciones

trabajador Examinar Estructura

Relaciones

Columna	Relación interna	Restricción de clave foránea (INNODB)
nombre	empresa.persona.nombre	'empresa'.`persona`.`nombre'
compañía	empresa.empresacompañía	¡No se ha definido ningún índice! Cree uno más abajo
salario		¡No se ha definido ningún índice! Cree uno más abajo

director_empresa Examinar Estructura

Relaciones

Columna	Relación interna	Restricción de clave foránea (INNODB)
compañía	empresa.empresacompañía	'empresa'.`empresa`.`compañía'
nombre_director	empresa.trabajadornombre	¡No se ha definido ningún índice! Cree uno más abajo

Creación de tablas: Ejercicios

PC (#modelo, velocidad, ram, hd, cd, precio)

Producto (#fabricante, #modelo, tipo)

Sabiendo que:

- ▶ El precio del PC por defecto será de 500.
- ▶ El tipo de producto sólo puede ser ‘p’ para ‘PC’ y ‘o’ para ‘otros’
- ▶ Los PC pueden no tener RAM
- ▶ Los valores para el CD será SI o NO para indicar si lleva o no lector de CDs

Creación de tablas: Ejercicios

Equipo (nombre, ciudad)

Jugador (DNI, nombre, dorsal, equipo)

Partido (id partido, fecha, resultado)

Juega (DNI, partido, posición)

Rivales (id partido, local, visitante)

Conjunto de tablas para trabajar

»»

Conjunto de tablas

- ▶ **Alumnos** (DNI, nom_alum, fecha_nac, provincia, beca)
- ▶ **Asignaturas** (cod_asig, nom_asig, creditos, curso)
- ▶ **Matriculas** (cod_asig, DNI, convocatoria, calificacion)
- ▶ **Profesores** (NRP, nom_prof, categoria, area, cod_dpto)
- ▶ **Aula** (cod_aula, capacidad)
- ▶ **Grupos** (cod_gru, cod_asig, cod_aula, tipo, NRP, max_al)

Conjunto de tablas

► Alumnos (DNI, nom_alum, fecha_nac, provincia, beca)

- DNI: Cadena de texto de 9 posiciones
- nom_alum: Cadena de texto 20 posiciones
- fecha_nac: fecha
- Provincia: cadena de texto de 15 posiciones
- Beca: solo puede valer ‘si’, ‘no’

Conjunto de tablas

► Asignaturas (cod_asig, nom_asig, creditos, curso)

- cod_asig: Autonumerico
- nom_asig: cadena de texto de 15 posiciones
- Creditos: numero entero de dos digitos. Puede estar vacío
- Curso: solo puede ser 1, 2 o 3

Conjunto de tablas

► **Matriculas** (cod_asig, DNI, convocatoria, calificacion)

- cod_asig: código de la asignatura en la que se matricula
- DNI: DNI del alumno que se matricula
- Convocatoria: número entero. Por defecto 1
- Calificacion: número de dos enteros y un decimal

Conjunto de tablas

▶ Profesores (NRP, nom_prof, categoria, area, cod_dpto)

- NRP: número entero de dos dígitos
- nom_prof: cadena de texto de 10 caracteres
- Categoria: solo puede ser ‘titular’ o ‘suplente’
- Area: cadena de texto de 15 posiciones
- Cod_dpto: entero de 2 digitos

Conjunto de tablas

► Aula (cod_aula, capacidad)

- cod_aula: cadena de texto de 3 dígitos
- Capacidad: número entero de 3 dígitos

Conjunto de tablas

► Grupos (cod_gru, cod_asig, cod_aula, tipo, NRP, max_al)

- Cod_gru: cadena de texto de 3 posiciones
- Cod_asig: código de la asignatura que se imparte en ese grupo
- Cod_aula: código del aula asignada al grupo
- Tipo: solo puede ser M si el grupo es de mañana y T si el grupo es de tarde
- NRP: código del profesor que imparte clase en el grupo
- max_al: número entero de 3 dígitos

Conjunto de tablas

- ▶ **Alumnos** (DNI, nom_alum, fecha_nac, provincia, beca)
- ▶ **Asignaturas** (cod_asig, nom_asig, creditos, curso)
 - ↑
- ▶ **Matriculas** (cod_asig, DNI, convocatoria, calificacion)
 - ↑
- ▶ **Profesores** (NRP, nom_prof, categoria, area, cod_dpto)
 - ↑
- ▶ **Aula** (cod_aula, capacidad)
 - ↑
- ▶ **Grupos** (cod_gru, cod_asig, cod_aula, tipo, NRP, max_al)

Inserción de filas

»

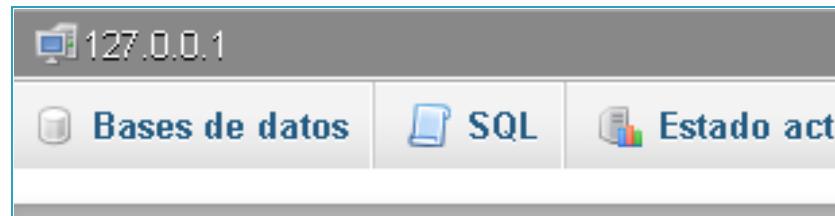
Inserción de filas

- ▶ La inserción de filas en MySQL se podría hacer igualmente con varios clic
- ▶ Nosotros necesitamos aprender la sentencia SQL puesto que será lo que utilizaremos en PHP para insertar nuevos datos
- ▶ Para insertar datos en una tabla, la sintaxis es sencilla:

```
INSERT INTO <nombre_tabla> VALUES  
(<valor1>, .....,<valorN>);
```

Inserción de filas en MySQL

- ▶ Para ejecutar la sentencia que inserta una fila en MySQL tenemos que ir a la sección SQL



- ▶ Se abrirá un cuadro donde podremos escribir la sentencia:

Inserción de filas en MySQL

Ejecute la o las consultas SQL en el servidor "127.0.0.1": [?](#)

```
1
```

Guardar esta consulta en favoritos:

[Delimitador :] Mostrar esta consulta otra vez Mantener la caja de texto con la consulta

Inserción de filas

- ▶ *Ejemplo: Insertar un nuevo alumno en la base de datos con los siguientes datos.*
 - *DNI = 44444444*
 - *Nombre: Carmen Garrido*
 - *Fecha de nacimiento: 28/09/75*
 - *Provincia: Granada*
 - *Beca: No*

ALGUNAS ACLARACIONES

- ▶ Datos de tipo **cadena** y **fecha** van entrecomillados.
- ▶ Los valores de tipo fecha suelen expresarse como cadenas, pero el formato específico depende de SGBD concreto.
 - En MySQL ‘aaaa-mm-dd’
- ▶ Los valores han de suministrarse en el mismo orden en el que están creados en la tabla.
- ▶ Si nos interesa dar un valor nulo a un campo concreto, podemos utilizar la palabra reservada **null** o dejar un espacio vacío en la posición que le corresponda.

Ejercicios

DNI	Nom_alum	Fecha_nac	Provincia	beca
11111111Z	Lucía	12/05/1992	Granada	No
22222222B	Mónica	18/12/1998	Jaén	Si
12345678C	Luis	03/01/1995	Granada	Si
33333333R	César	08/09/1993	Granada	No
55555555T	Roberto	24/11/1998	Málaga	Si

Cod_asig	Nom_asig	Creditos	curso
1	Lengua	14	1
2	Matemáticas	8	1
3	Ciencias	null	2
4	Literatura	7	2
5	Historia	null	1
6	Dibujo	12	2

Ejercicios

Cod_asig	DNI	Convocatoria	Calificación
1	11111111Z	1	3
1	12345678C	1	8
2	11111111Z	1	5
2	55555555T	1	7
3	11111111Z	1	7
4	12345678C	1	3
2	33333333R	1	4
1	11111111Z	2	6
1	12345678C	2	2

NRP	Nom_prof	Categoría	Area	Cod_depto
21	Alfonso	Titular	Ciencias	1
34	Helena	Titular	Ciencias	2
15	Francisco	Suplente	Lengua	3

Cod_aula	Capacidad
A1	25
A2	20
A3	22

Cod_gru	Cod_asig	cod_aula	tipo	NRP	Max_al
G1	2	A1	M	21	23
G2	1	A2	M	15	18
G3	3	A3	M	21	20
G4	3	A3	T	15	20

Eliminación de filas

»

Eliminación de filas

- ▶ El borrado de filas es muy sencillo en SQL. La sentencia es:

```
DELETE FROM <nombre_tabla>;
```

- ▶ *Ejemplo: Borrar todas las filas de la tabla alumnos.*

```
DELETE FROM alumnos;
```

Eliminación de filas

- ▶ No siempre nos interesa borrar todas las filas de una tabla.
- ▶ SQL incluye una segunda cláusula dentro de la sentencia DELETE
- ▶ **WHERE** permite incluir la condición que deben cumplir las filas que se borrarán:

```
DELETE FROM <nombre_tabla>
```

```
WHERE condición;
```

Eliminación de filas

- ▶ *Ejemplo: Borrar todas las filas de los alumnos que tienen beca.*

```
DELETE FROM alumnos WHERE beca='SI';
```

- ▶ *Ejemplo: Borrar todas las asignaturas de 2º curso*

```
DELETE FROM asignaturas WHERE curso=2;
```

- ▶ *Ejemplo: Borrar todos los alumnos que vienen de Granada*

```
DELETE FROM alumnos WHERE provincia='Granada'
```

Eliminación de filas: Ejercicios

- ▶ Eliminar de la base de datos los siguientes elementos:
 - Borrar todos los alumnos de Jaén
 - Borrar todas las asignaturas con menos de 10 créditos
 - Borrar a todos los profesores del departamento 3
 - Borrar los grupos con una capacidad inferior a 25 alumnos

Actualización de filas

»»

Actualización de filas

- ▶ Se realiza mediante el uso de la sentencia **UPDATE**.
- ▶ La sintaxis es la siguiente:

```
UPDATE <nombre_tabla>
SET <nombre_columna>=<nuevo_valor>
[{,<nombre_columna>=<nuevo_valor>}]
[WHERE condicion];
```

Ejemplos

- ▶ *Ejemplo: Cambiar el nombre del alumno con DNI=44444444 por el de JUAN LOPEZ*

```
UPDATE alumnos  
SET nombre='Juan López'  
WHERE DNI = '44444444';
```

- ▶ *Ejemplo: Subir un punto la nota del alumno de DNI '44444444' en la asignatura de código BD2 y aumentarle una convocatoria*

```
UPDATE matriculas  
SET calificación = calificación + 1,  
    convocatoria = convocatoria + 1  
WHERE DNI = '44444444' and cod_asig = 'BD2';
```

Actualización de filas: Ejercicios

- ▶ Cambiar de la base de datos los siguientes datos:
 - La alumna Lucía se ha mudado a Málaga
 - A todos los alumnos se les baja la nota 1 punto
 - El profesor con código 34 pasa a ser Suplente
 - Aquellas aulas con capacidad entre 20 y 23, subirles la capacidad en 10 alumnos más
 - El profesor con código 21 pasa a ser Suplente y además del departamento 3

Manipulando datos con SQL

»»

Manipulación de datos

- ▶ Para recuperar los datos almacenados en las tablas de nuestra base de datos, SQL dispone de la sentencia **SELECT**.
- ▶ La sintaxis básica de esta sentencia tiene la siguiente forma:

```
SELECT <nombre_coluna> [{,nombre_columna}]  
FROM <nombre_tabla>  
[WHERE <condicion>];
```

- ▶ Hay tres cláusulas principales:
 - **SELECT** contiene las columnas que queremos mostrar.
 - **FROM** indica sobre qué tabla queremos consultar.
 - **WHERE** impone una condición booleana que deben cumplir las tuplas para ser recuperadas.

Consultas Sencillas

Mostrar sólo los nombres de los alumnos.

▶ El proceso a seguir para conseguirlo:

- Decidir qué tabla o tablas necesitamos:
 - Alumnos
- Decidir qué columna o columnas queremos mostrar:
 - Nom_alum
- Construir la sentencia SQL:

```
SELECT nom_alum
```

```
FROM alumnos;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

Nom_alum
Lucía
Mónica
Luis
César
Roberto

Consultas Sencillas

Mostrar la provincia de los alumnos

- Decidir qué tabla o tablas necesitamos:
 - alumnos
- Decidir qué columna o columnas queremos mostrar:
 - provincia
- Construir la sentencia SQL:

```
SELECT provincia  
FROM alumnos;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

Provincia
Granada
Jaén
Granada
Granada
Málaga

Consultas Sencillas

Mostrar el DNI y el nombre de los alumnos

- Decidir qué tabla o tablas necesitamos:
 - alumnos
- Decidir qué columna o columnas queremos mostrar:
 - DNI, nom_alum
- Construir la sentencia SQL:

```
SELECT DNI, nom_alum  
FROM alumnos;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

DNI	Nom_alum
11111111Z	Lucía
22222222B	Mónica
12345678C	Luis
33333333R	César
55555555T	Roberto

Consultas Sencillas

Mostrar todos los datos de los alumnos

- Decidir qué tabla o tablas necesitamos:
 - alumnos
- Decidir qué columna o columnas queremos mostrar:
 - Todas
- Construir la sentencia SQL:

```
SELECT DNI, nom_alum, fecha_nac, provincia, beca  
FROM alumnos;
```

```
SELECT *  
FROM alumnos;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

DNI	Nom_alum	Fecha_nac	Provincia	beca
11111111Z	Lucía	12/05/1992	Granada	No
22222222B	Mónica	18/12/1998	Jaén	Si
12345678C	Luis	03/01/1995	Granada	Si
33333333R	César	08/09/1993	Granada	No
55555555T	Roberto	24/11/1998	Málaga	Si

Consultas Sencillas

Mostrar el nombre de los alumnos que son de Granada

- Decidir qué tabla o tablas necesitamos:
 - alumnos
- Decidir qué columna o columnas queremos mostrar:
 - Nom_alum
- Decidir qué condición o condiciones deben cumplir las filas mostradas:
 - Provincia = ‘Granada’
- Construir la sentencia SQL:

```
SELECT nom_alum  
FROM alumnos  
WHERE provincia = ‘Granada’;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

Nom_alum
Lucía
Luis
César

Aclaración

Las condiciones de la cláusula WHERE pueden ser tan complicadas como el usuario desee.

- Identificadores de columnas
- Literales
- Operadores de comparación (<, >, >=, <=, <>, =)
- Operadores lógicos (AND, OR, NOT)

Consultas Sencillas

Mostrar el nombre de los alumnos que tienen beca

- Decidir qué tabla o tablas necesitamos:
 - alumnos
- Decidir qué columna o columnas queremos mostrar:
 - Nom_alum
- Decidir qué condición o condiciones deben cumplir las filas mostradas:
 - beca = ‘Si’
- Construir la sentencia SQL:

```
SELECT nom_alum  
FROM alumnos  
WHERE beca= ‘Si’;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

Nom_alum
Mónica
Luis
Roberto

Consultas Sencillas

Mostrar el nombre y DNI de los alumnos que NO son de Granada

- Decidir qué tabla o tablas necesitamos:
 - alumnos
- Decidir qué columna o columnas queremos mostrar:
 - Nom_alum, DNI
- Decidir qué condición o condiciones deben cumplir las filas mostradas:
 - Provincia <> ‘Granada’
- Construir la sentencia SQL:

```
SELECT nom_alum, DNI  
FROM alumnos  
WHERE provincia <> ‘Granada’;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

Nom_alum	DNI
Mónica	22222222B
Roberto	55555555T

Consultas Sencillas

Mostrar todos los datos del alumno llamado Lucía

- Decidir qué tabla o tablas necesitamos:
 - alumnos
- Decidir qué columna o columnas queremos mostrar:
 - Todos
- Decidir qué condición o condiciones deben cumplir las filas mostradas:
 - Nom_alum= ‘Lucía’
- Construir la sentencia SQL:

```
SELECT *
FROM alumnos
WHERE nom_alum= ‘Lucía’;
```

Consultas Sencillas

- ▶ El resultado de la ejecución de esta sentencia es el siguiente:

DNI	Nom_alum	Fecha_nac	Provincia	beca
11111111Z	Lucía	12/05/1992	Granada	No

Consultas Sencillas

- ▶ Supongamos que el contenido de la tabla asignaturas es el siguiente:

Cod_asig	Nom_asig	Creditos	curso
1	Lengua	14	1
2	Matemáticas	8	1
3	Ciencias	19	2
4	Literatura	7	2
5	Historia	25	1
6	Dibujo	12	2

Consultas sencillas

- ▶ Realicemos las siguientes consultas:
 - Mostrar el nombre de todas las asignaturas.
 - Mostrar lo créditos de todas las asignaturas.
 - Mostrar código de asignatura y curso de todas las asignaturas.
 - Mostrar los créditos de las asignaturas de 2º curso.
 - Mostrar todos los datos de la asignatura con código 4
 - Mostrar todos los datos de aquellas asignaturas con más de 15 créditos.
 - Mostrar los nombres de las asignaturas que tienen entre 12 y 25 créditos.

Consultas sencillas

- ▶ Realicemos las siguientes consultas:
 - Mostrar el DNI de los alumnos matriculados.
 - Mostrar el DNI de los alumnos matriculados en la asignatura 3.
 - Mostrar el DNI de los alumnos con algo suspenso.
 - Mostrar el DNI de los alumnos que han aprobado la asignatura con código 1.
 - Mostrar el DNI y el código de la asignatura de los alumnos que hayan sacado en algo más de un 7.
 - Mostrar el DNI de los alumnos con asignaturas aprobadas.

Anotaciones



Eliminación de valores duplicados

DNI	Nom_alum	Fecha_nac	Provincia	beca
11111111Z	Lucía	12/05/1992	Granada	No
22222222B	Mónica	18/12/1998	Jaén	Si
12345678C	Luis	03/01/1995	Granada	Si
33333333R	César	08/09/1993	Granada	No
55555555T	Roberto	24/11/1998	Málaga	Si

Eliminación de valores duplicados

Ejemplo: Recuperar las ciudades de las que provienen los Alumnos.

```
SELECT provincia  
      FROM alumnos;
```

Esta consulta daría como resultado lo siguiente:

Provincia
Granada
Jaén
Granada
Granada
Málaga

Eliminación de valores duplicados

- ▶ Con SQL, si queremos eliminar las filas duplicadas del resultado debemos solicitarlo mediante **DISTINCT**

```
SELECT DISTINCT provincia  
FROM alumnos;
```

- ▶ El resultado en este caso sería:

Provincia
Granada
Jaén
Málaga

Ordenación de los resultados

- ▶ SQL, al ejecutar una consulta, muestra los datos en el mismo orden en el que se encuentran en la tabla original.
- ▶ Sin embargo, nosotros podemos indicar que queremos que los resultados se muestren en otro orden.
- ▶ Utilizamos para ello la cláusula **ORDER BY**.
ORDER BY <nombre_atributo>
- ▶ La cláusula ORDER BY admite además indicar el tipo de ordenación.
 - **ASC**
 - **DESC**

Ordenación de los resultados

Ejemplo: Mostrar la lista de los becarios ordenados en orden alfabético.

```
SELECT *
FROM alumnos
WHERE beca='SI'
ORDER BY nom_alum ASC;
```

DNI	Nom_alum	Fecha_nac	Provincia	beca
12345678C	Luis	03/01/1995	Granada	Si
22222222B	Mónica	18/12/1998	Jaén	Si
55555555T	Roberto	24/11/1998	Málaga	Si

Ordenación de los resultados

Ejemplo: Mostrar la lista de los alumnos ordenados por su provincia de procedencia de forma descendente y dentro de cada provincia, ordenados alfabéticamente.

```
SELECT *
FROM alumnos
ORDER BY provincia DESC, nom_alum ASC;
```

Provincia	Nom_alum
Málaga	Roberto
Jaén	Mónica
Granada	César
Granada	Lucía
Granada	Luis

Consultas sobre varias tablas



Consultas sobre varias tablas

- ▶ Hay veces que es necesario utilizar datos que están contenidos en distintas tablas.
- ▶ En SQL, se pueden incluir varias tablas en la cláusula **FROM**.
- ▶ En este caso el sistema hace el **producto cartesiano** de todas las tablas incluidas y luego realiza la consulta sobre la tabla resultado de ese producto cartesiano.
- ▶ Esto quiere decir que tenemos que eliminar las **filas no reales**.

Consultas sobre varias tablas

- ▶ Para explicar esto, vamos a utilizar dos nuevas tablas:
- ▶ Dueños:
- ▶ Perros:

DNI	Nom_dueño
111111S	Roció
333333E	Paloma
666666B	Víctor

Num_perro	Nom_perro	Dni_dueño
1	Ali	111111S
2	Buda	333333E
3	Pico	111111S
4	Rufo	666666B

Consultas sobre varias tablas

- ▶ Supongamos que nos piden mostrar los perros de cada uno de los dueños.
- ▶ La consulta:

```
Select *
```

```
From dueños, perros;
```

- ▶ Dará como resultado:

Consultas sobre varias tablas

DNI	Nom_dueño	Num_perro	Nom_perro	DNI_dueño
111111S	Roció	1	Ali	111111S
111111S	Roció	2	Buda	333333E
111111S	Roció	3	Pico	111111S
111111S	Roció	4	Rufo	666666B
333333E	Paloma	1	Ali	111111S
333333E	Paloma	2	Buda	333333E
333333E	Paloma	3	Pico	111111S
333333E	Paloma	4	Rufo	666666B
666666B	Víctor	1	Ali	111111S
666666B	Víctor	2	Buda	333333E
666666B	Víctor	3	Pico	111111S
666666B	Víctor	4	Rufo	666666B

Consultas sobre varias tablas

- ▶ Dentro de esta tabla, hay muchas filas que **no son “REALES”** ya que no unen de verdad a cada perro con su dueño:

Consultas sobre varias tablas

DNI	Nom_dueño	Num_perro	Nom_perro	DNI_dueño
111111S	Roció	1	Ali	111111S
111111S	Roció	2	Buda	333333E
111111S	Roció	3	Pico	111111S
111111S	Roció	4	Rufo	666666B
333333E	Paloma	1	Ali	111111S
333333E	Paloma	2	Buda	333333E
333333E	Paloma	3	Pico	111111S
333333E	Paloma	4	Rufo	666666B
666666B	Víctor	1	Ali	111111S
666666B	Víctor	2	Buda	333333E
666666B	Víctor	3	Pico	111111S
666666B	Víctor	4	Rufo	666666B

Consultas sobre varias tablas

- ▶ ¿Qué pasa con las filas que SI son reales? ¿Qué tienen en común todas ellas?
 - DNI = DNI_dueño
- ▶ Por tanto, una vez unidas las tablas, deberemos quedarnos solo con las filas que nos interesan:

```
Select *  
From dueños, perros  
Where dni = dni_dueño;
```

DNI	Nom_dueño	Num_perro	Nom_perro	DNI_dueño
111111S	Roció	1	Ali	111111S
111111S	Roció	3	Pico	111111S
333333E	Paloma	2	Buda	333333E
666666B	Víctor	4	Rufo	666666B

Consultas sobre varias tablas

Volvamos ahora a las tablas que teníamos...

Consultas sobre varias tablas

Mostrar para cada matricula, el nombre de la asignatura a la que corresponde.

- ▶ Tablas que necesitamos:
 - Matriculas, asignaturas.
- ▶ Columnas que queremos visualizar:
 - Todas las de matriculas, nombre de asignatura.
- ▶ Condición que deben cumplir las filas visualizadas:
 - Matriculas.cod_asig = asignaturas.cod_asig

Consultas sobre varias tablas

▶ Consulta:

```
Select matriculas.*, asignaturas.nom_asig  
From matriculas, asignaturas  
Where matriculas.cod_asig = asignaturas.cod_asig;
```

Cod_asig	DNI	Convocatoria	Calificación	Nom_asig
1	11111111Z	1	3	Lengua
1	12345678C	1	8	Lengua
2	11111111Z	1	5	Matemáticas
2	55555555T	1	7	Matemáticas
3	11111111Z	1	7	Ciencias
4	12345678C	1	3	Literatura
2	33333333R	1	4	Matemáticas
1	11111111Z	2	6	Lengua
1	12345678C	2	2	Lengua

Consultas sobre varias tablas

- ▶ Mostrar la tabla matriculas, incluyendo el nombre de cada alumno.
- ▶ Mostrar el nombre de las asignaturas en las que está matriculado cada alumno.
- ▶ Mostrar el nombre de aquellos alumnos con más de un 4 en la asignatura con código 2.
- ▶ Visualizar el nombre de las asignaturas con alumnos suspensos.
- ▶ Mostrar todos los datos de las asignaturas en las que está matriculado el alumno llamado “Mónica”

Consultas sobre varias tablas

- ▶ Mostrar el nombre de los profesores que dan clase en el aula A3.
- ▶ Mostrar los códigos de los grupos en los que se imparte Ciencias.
- ▶ Mostrar los códigos de los grupos en los que imparte clase el profesor Francisco.
- ▶ Mostrar la capacidad de las aulas en las que se imparte Lengua.
- ▶ Listar el nombre y la categoría de los profesores que dan clase en el grupo G2.
- ▶ Visualizar todos los datos de los grupos en los que imparte clase Helena

Consultas sobre varias tablas

- ▶ Mostrar los nombres y las categorías de los profesores que imparten la asignatura de “Ciencias”.
- ▶ Visualizar los nombres de las asignaturas que se imparten por la mañana.
- ▶ Mostrar los nombres de las asignaturas que imparten los profesores del departamento 2.
- ▶ Mostrar los nombres de las asignaturas que tienen aprobadas los alumnos de Granada.
- ▶ Listar las provincias de las que vienen los alumnos con más de un 7 en matemáticas.

Consultas sobre varias tablas

- ▶ Mostrar los nombres y las notas de todos los alumnos de Málaga, ordenados alfabéticamente.
- ▶ Mostrar las provincias de los alumnos que han sacado un 7 en Matemáticas.
- ▶ Mostrar las aulas en las que se da clase de asignaturas que aun no tienen asignados créditos.
- ▶ Mostrar los nombres y las áreas de los profesores, que imparten clase por la mañana.
- ▶ Visualizar una lista de los grupos en los que imparten clase los profesores del área de Ciencias.
- ▶ Listar los datos de los alumnos que cursan Matemáticas, ordenados por su fecha de nacimiento.

Sistema Gestor de Bases de Datos

» MySQL

Pasos a seguir para trabajar con Bases de Datos desde PHP

Paso 1: Conectar con el servidor de Bases de Datos

Función: mysqli_connect

Recibe:

- Nombre del servidor
- Nombre de usuario
- Contraseña de usuario
- Nombre de la base de datos elegida

Devuelve:

- conector de la Base de Datos si todo ha ido bien
- FALSE si ha habido algún error

Ejemplo:

```
$conexion = mysqli_connect ('localhost', 'root', '', 'centro');
```

Paso 2: Construcción de la sentencia SQL a ejecutar

Sentencia: debe estar dentro de una cadena de texto

Ejemplo:

```
$consulta = "select * from clientes";
```

Paso 3: Ejecutar la sentencia en la base de datos

Función: mysqli_query

Recibe:

- Conexión con el servidor
- Sentencia que queremos ejecutar

Devuelve:

- FALSE se ha habido algún error
- Si la sentencia era insert, delete o update: TRUE si todo ha ido bien.
- Si la sentencia era select: conjunto de valores resultado de la consulta

Ejemplo:

```
$resul = mysqli_query($conexion, $consulta);
```

Paso 4: Comprobación de errores

Función: mysqli_error

Recibe:

- Conector del servidor

Devuelve:

- Cadena con el error que ha habido.
- FALSE si no ha habido errores de ningún tipo

Ejemplo:

```
$error = mysqli_error($conexion);
```

Función: mysqli_error_list

Recibe:

- Conector del servidor

Devuelve:

- Array con los errores que ha habido.
- FALSE si no ha habido errores de ningún tipo

Ejemplo:

```
$datos = mysqli_query($conexion, $consulta);
```

Paso 5: Trabajar con los datos devueltos por el select

Función: mysqli_fetch_array

Recibe:

- Conjunto de datos devuelto por la consulta select

Devuelve:

- Array asociativo con el contenido de una de las filas del select. El array tiene dos posiciones por cada columna que se visualice en el select
 - Una:
 - ✓ Como nombre: nombre de la columna
 - ✓ Como contenido: el contenido de esa columna en la fila correspondiente
 - Otra:
 - ✓ Como nombre: numero de la columna
 - ✓ Como contenido: el contenido de esa columna en la fila correspondiente
- FALSE cuando ya no quedan más filas que consultar

NOTA: normalmente esta función se utiliza junto con bucles que se mueven mientras que haya datos que consultar.

Ejemplo:

```
while ($fila = mysqli_fetch_array($resul))
{
    /*Tratamiento de la información
    obtenida al realizar
    la consulta */
}
```

Paso 7: cerrar la conexión

Función: mysqli_close

Recibe:

- Conector que queremos cerrar

Devuelve:

- TRUE si la conexión se ha cerrado bien.
- FALSE si hay algún problema

Ejemplo:

```
mysqli_close($conexion);
```

Otras funciones para trabajar con Bases de Datos desde PHP

Comprobación del número de filas afectadas por una sentencia

Función: mysqli_affected_rows

Recibe:

- Conector de la base de datos

Devuelve:

- Número de filas que se han insertado, modificado o borrado o que han sido devueltas por un select

Ejemplo:

```
$num_registros = mysqli_affected_rows ($conexion);
```

MYSQLI

PHP y MySQLi



¿QUÉ ES MYSQLI?

Introducción

- MySQLi es el acrónimo de MySQL **improved** (MySQL mejorado)
- Las funciones de MySQL están en **desuso** desde la versión 4.1.3
- MySQLi se desarrolló para aprovechar las nuevas **funcionalidades** encontradas en los sistemas MySQL desde esa versión 4.1.3

Introducción

- La extensión MySQLi contiene muchos beneficios, entre los que se encuentran:
 - ✓ Interfaz orientada a objetos
 - ✓ Soporte para Declaraciones Preparadas
 - ✓ Soporte para Múltiples Declaraciones
 - ✓ Soporte para Transacciones
 - ✓ Mejoradas las opciones de depuración
 - ✓ Soporte para servidor empotrado

CONEXIÓN CON EL SERVIDOR

Conexión con el servidor

- Lo primero que hay que hacer es **crear** una conexión desde el documento PHP hasta el Sistema Gestor de Bases de Datos
- Al crear la conexión se genera un **canal** a través del cual se van a enviar todas las consultas y todas las peticiones a la Base de Datos

Conexión con el servidor

- **mysqli_connect(servidor, usuario, pass, bd)**
- **Servidor:** Nombre del servidor al que nos queremos conectar (localhost)
- **Usuario:** Nombre del usuario con el que nos vamos a conectar (root)
- **Pass:** Contraseña del usuario con el que nos conectamos al servidor (vacia)
- **BD:** Base de datos a la que queremos conectarnos de todas las que haya en el servidor
- Devuelve un conector (canal) que debemos recoger en una variable.

REALIZACIÓN DE CONSULTAS

Realización de consultas

- La consulta que queramos enviar a la base de datos será una **cadena de texto**.
- Puede estar almacenada en una variable o no
- Podemos realizar **cualquier operación** de las que hemos visto en SQL
- Para ejecutar una consulta debemos utilizar la función:

Realización de consultas

- **mysqli_query (conector, consulta)**
 - **conector:** canal creado al hacer mysqli_connect
 - **consulta:** cadena de texto con la operación que queremos ejecutar.
- Dependiendo del tipo de operación que se ejecute el resultado de mysqli_query va a ser diferente

```
$datos = mysqli_query($conector, $consulta)
```

Resultado de *mysqli_query*

- Si la consulta es un **SELECT**:
 - ✓ FALSE si ocurre un error.
 - ✓ Conjunto de datos si todo va bien.
- Si la consulta era **distinta de SELECT** (insert, update, delete):
 - ✓ FALSE si ocurre un error.
 - ✓ TRUE si todo va bien.

MANIPULACIÓN DE LOS RESULTADOS

Manipulación de los resultados

- Una vez ejecutada la consulta, podemos saber a cuántas filas a afectado dicha consulta:
- **mysqli_affected_rows (conector)**
 - Devuelve el número de filas afectadas por la última consulta realizada de tipo **insert, update o delete**
 - Devuelve el número de filas devueltas por la última consulta realizada de tipo **select**

Manipulación de los resultados

- Normalmente una sentencia Select devuelve un **conjunto de valores**
- Este conjunto de valores **no** es accesible directamente
- Hay que ir recogiendo fila a fila y trabajando con ellas de forma independiente

Manipulación de los resultados

➤ `mysqli_fetch_array (datos)`

- Devuelve una **array asociativo** en el que encontraremos una de las filas del conjunto de datos.
- El array tiene por cada campo devuelto por el SELECT **dos** posiciones.
 - ✓ *Numérica*: 0, 1, 2...
 - ✓ *Asociativa*: Nombre del campo devuelto
- Devuelve **NULL** cuando no hay más datos para mostrar

Ejemplo

```
$consulta = "select nombre, edad from alumnos";  
$resul = mysqli_query($conexion, $consulta);  
  
$num_campos = mysqli_num_rows($resul);  
$fila=mysqli_fetch_array($resul)
```

- En este caso el array **\$fila** tendrá la siguiente estructura:

0	Nombre	1	Edad
Juan	Juan	24	24

Manipulación de los resultados

➤ **mysqli_fetch_assoc (datos)**

- Devuelve un **array asociativo** en el que encontraremos una de las filas del conjunto de datos.
- En este caso el array **sólo tiene una posición** por cada campo devuelto por el select
- El nombre de la posición será el nombre del campo devuelto por select
- Devuelve **Null** cuando no hay más datos

Manipulación de los resultados

➤ **mysqli_fetch_all (datos)**

- Devuelve en este caso una **matriz posicional** que tendrá:
 - ✓ Tantas filas como filas devuelva la consulta select
 - ✓ Tantas columnas como campos le pidamos a la consulta select.

Manipulación de los resultados

```
$consulta = "select nombre, edad from alumnos";  
$datos = mysqli_query($conector, $consulta);  
$matriz = mysqli_fetch_all($datos);
```

	0	1
0	Ramón Torres	19
1	María López	21
2	Paloma Ruiz	24
3	Isabel Perea	25
4

COMPROBACIÓN DE ERRORES

Comprobación de errores

➤ **mysqli_error (conector)**

- Devuelve el mensaje de error para la última acción que se haya hecho por el conector
- Lo que devuelve es una cadena de texto

Comprobación de errores

➤ Ejemplo

```
$consulta = "select nommbre from alumnos";  
  
$datos = mysqli_query($conexion, $consulta);  
  
echo mysqli_error($conexion);
```

Unknown column 'nommbre' in 'field list'

Comprobación de errores

➤ **mysqli_error_list (conector)**

- Devuelve en este caso una **matriz** en la que se almacenarán una lista con los errores que se hayan producido.
- Cada error será una fila de la matriz que tendrá:
- **Errno**: número del error ocurrido
- **Error**: cadena de texto asociada al error
- **Sqlstate**: error con la nomenclatura SQLSTATE

Comprobación de errores

➤ Ejemplo

```
$conexion = mysqli_connect("localhost", "root", "", "cenro");

$consulta = "select nommbre from alumnos";
$datos = mysqli_query($conexion, $consulta);

print_r(mysqli_error_list($enlace));
```

```
Array
(
    [0] => Array
        (
            [errno] => 1054
            [sqlstate] => 42S22
            [error] => Unknown column 'nommbre' in 'field list'
        )
)
```



CERRAR LA CONEXIÓN



Cerrar la conexión

- Una vez hecho todo lo necesario es muy importante cerrar la conexión con la Base de Datos
- Mientras la conexión esté abierta nadie más podrá usar la Base de Datos
- **mysqli_close(conector)**
 - Cierra la conexión abierta

EJEMPLO

Ejemplo

- Vamos a conectarnos a la base de datos CENTRO para mostrar el nombre de todos sus alumnos
- Para ello:

Ejemplo

- 1º Abrimos la conexión y elegimos la Base de datos a la que nos queremos conectar:

```
$conexion = mysqli_connect ('localhost', 'root', '', 'centro');  
if (!$conexion)  
{  
    echo "<h3> ~~~~~~ERROR AL CONECTAR CO EL SGBD~~~~~</h3>";  
}
```



Ejemplo

- 2º Escribimos la consulta que queremos ejecutar:

```
$consulta = "select nom_alum from alumnos";
```

Ejemplo

- 3º Ejecutamos la consulta y comprobamos el número de registros que ha devuelto

```
$resul = mysqli_query($conexion, $consulta);  
$num_registros = mysqli_affected_rows ($conexion);
```

Ejemplo

- 4º Mostramos por pantalla los datos que hemos obtenido

```
echo "Hay un total de $num_registros alumnos en el centro<br>";
echo "Nombre de los alumnos: <br>";

while ($fila = mysqli_fetch_array($resul))
{
    echo "<br>".$fila['nom_alum'];
}
```

Ejemplo

- 5º y último: cerramos la conexión con la base de datos

```
mysqli_close($conexion);
```

Ejemplo

Hay un total de 5 alumnos en el centro
Nombre de los alumnos:

Lucia
Luis
Monica
Cesar
Roberto

CONSEJOS

Consejos

- Puesto que ciertas acciones las realizaremos siempre igual:
 - ✓ Abrir conexión
 - ✓ Comprobar que todo ha ido bien
 - ✓ Seleccionar la base de datos
 - ✓ Etc
- Lo mejor es colocarlo en funciones a la que iremos llamando cuando sea necesario

MYSQLI

PHP y MySQLi

