Curso de programación en PHP con MySQL

(PHP Versión 5.1.2, MySQL versión 5.0.24)

Manual del alumno

Ing. Cedric Simon – Tel: 268 09 74 – Cel: 888 23 87 – Email: cedric@solucionjava.com – Web: www.solucionjava.com

SolucionJava.com

Autor: Cédric Simon

1. Índice

2. Introducción al curso

2.1. Objetivo de este curso

En este curso vamos a ver el lenguaje PHP y como conectarse a una base de datos de tipo MySQL de desde PHP.

2.2. Manual del alumno

Este manual del alumno es una ayuda para el alumno, para tenga un recuerdo del curso. Este manual contiene un resumen de las materias que se van a estudiar durante el curso, pero el alumno debería de tomar notas personales para completas este manual.

En el CD de curso viene tambien la documentación oficial de PHP y de MySQL en formato HTML (debajo la carpeta 'documentacion'). Esta documentación completa este manual.

Este manual del usuario viene tambien en formato PDF en el CD del curso.

2.3. Requisitos para atender a este curso

El conocimiento del lenguaje HTML es requerido para poder seguir este curso.

3. Introducción al PHP

3.1. ¿Que es el PHP?

El PHP es un lenguaje de programación utilizado para crear páginas web dinámicas.

El PHP necesita que un servidor web con capacidad PHP sea instalado y funcionando para poder ejecutar las páginas PHP.

El servidor va a compilar el código PHP y tratarlo en tiempo real, con la información viniendo del cliente web, para regresarle una pagina web adaptada, en tiempo real.

El servidor tiene también capacidad de seguir el camino de un usuario, así cómo de identificarlo.

3.2. ¿Por qué utilizaríamos el PHP?

El PHP se necesita cuando la pagina web tiene que adaptarse en función del usuario, y cuando se necesita guardar valores de sesión del usuario.

Existen otros lenguaje que permiten eso, como el ASP o el JSP, pero el PHP tiene como aventaje que es un lenguaje fácil a aprender. Los servidores PHP (como los servidores JSP) existen para varios sistemas operativos, entre otros Windows, Linux, y Unix.

El PHP es un estándar de programación Internet.

3.3. Navegadores web

Si los navegadores prueban todos de poder aplicar al máximo las recomendaciones del HTML 4.0 existen ciertas opciones, a dentro o afuera del HML estándar que sólo ciertos navegadores soportan. También, un mismo código no aparecerá siempre igual en un navegador e en otro.

El PHP no está ligado directamente con los navegadores, pero el HTML que generaran si. Así que un código generado podría funcionar bien en un navegador, y dar un resultado diferente en otro.

En este curso, utilizaremos el navegador Firefox de Mozilla.

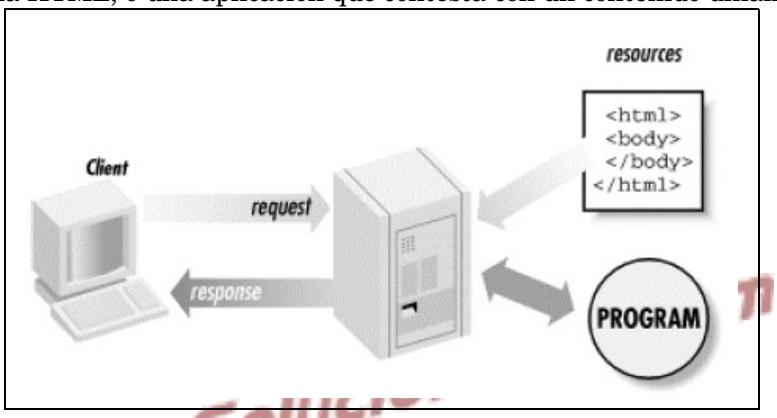
4. El protocolo HTTP

El protocolo HTTP (HyperText Transport Protocol) es un protocolo que define en detalle cómo un cliente y un servidor deben comunicar entre ellos. El modelo de comunicación HTTP es a la base del web. El protocolo se aplica a cualquier servidor y cliente web. Vamos a examinar los detalles más importante del protocolo HTTP.

4.1. Modelo de consulta / respuesta HTTP

El HTTP y los protocolos extendidos basados sobre el utilizan un modelo de comunicación simple, pero poderoso.

De manera típica, un cliente envía una consulta para un recurso a un servidor, y el servidor contesta una respuesta que corresponde al recurso preguntado (o contesta con un mensaje de error). El recurso puede ser una página HTML, o una aplicación que contesta con un contenido dinámico.



Este modelo simple implica tres cosas que Usted tiene que ser conciente:

1. El protocolo HTTP es un protocolo sin estado. Es decir que el servidor no guarda ningún información acerca del cliente depuse de haberle contestado, y por eso no puede reconocer que varias consultas de un mismo cliente pueden ser relacionadas.
2. La aplicación web no puede entregar una respuesta inmediato, como en aplicaciones locales. La velocidad depende del ancho de banda disponible y de la carga del servidor.
3. No hay nada en el protocolo que indica al servidor como la consulta le ha llegado, así que el servidor no puede distinguir diferente métodos de consulta. Por ejemplo, en servidor no puede distinguir una consulta generada por un clic en un enlace del uso del botón 'atrás' del navegador. También, como el HTTP es sin estado, no se puede a dentro del HTTP llamar a la página anterior.

4.2. Detalles de una consulta

Existen dos métodos de consulta: GET y POST. GET es la más utilizada, y la que se uso por defecto.

Ejemplo de una consulta:

```
GET /index.html HTTP/1.0
Host: www.gefionsoftware.com
User-Agent : Mozilla/4.5 [en] (WinNT; I)
Accept: image/gif, image/jpeg, image/pjpeg, image/png, */*
Accept-language : en
Accept-charset : iso-8859-1,*,utf-8
```

La primera línea especifica que se usa el método GET y se pregunta para regresar el recurso /index.html utilizando el protocolo HTTP/1.0. Las otras líneas proveen títulos con información adicional al servidor para cumplir con la consulta.

El título HOST dice al servidor el nombre (hostname) utilizado en el URL. Un servidor puede tener varios nombres, y este información permite distinguir múltiple servidores virtuales utilizando un mismo proceso web.

El título `User-Agent` contiene información sobre el tipo de navegador utilizado para hacer la consulta. El servidor puede utilizar este información para generar respuestas diferentes dependiendo del navegador (IE, Netscape, WAP, PDA,...).

Los títulos `Accept` proveen información sobre el idioma y el formato de archivo que el navegador acepta.

4.3. Detalles de respuesta

El mensaje de la respuesta parece al de la consulta. El contiene tres partes: una línea de estado, unos títulos de respuesta, y el cuerpo de la respuesta.

Ejemplo de respuesta:

```
HTTP/1.0 200 OK
Last-Modified: Mon, 19 Dec 2004 20:21:42 GMT
Date: Tue, 12 Jul 2005 13:12:10 GMT
Status: 200
Content-Type: text/html
Servlet-Engine: Tomcat Web Server/3.2
Content-Length: 59

<html>
<body>
<h1>Hello World!</h1>
</body>
</html>
```


La línea de estado empieza con el nombre del protocolo, seguido por el código de resultado y una breva descripción del código de resultado. Aquí el código de resultado es 200, que significa que salió con éxito.

El mensaje de respuesta tiene títulos, como el de consulta. En este ejemplo:

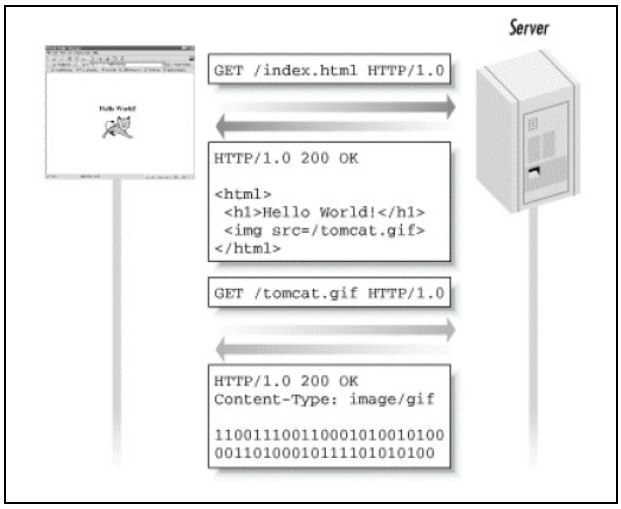
El título `Last-Modified` da la fecha y hora de cuando el recurso fue modificado por última vez. El cliente puede utilizar esta información para marcar la página el su caché, y pedir en el futuro solamente las que fueron modificadas.

El título `Content-Type` dice al navegador que tipo de datos contiene el cuerpo de la respuesta.

El título `Content-Length` dice al navegador el tamaño del cuerpo de la respuesta.

Una línea vacía separa los títulos del cuerpo de la respuesta. El cuerpo de la respuesta contiene el código que será enseñado en el navegador. Aquí una simple página HTML.

El cuerpo puede contener páginas HTML más complicadas, u otro tipo de contenido (imagen, sonido, archivo comprimido,...). El cuerpo puede también contener código que va generar otras consultas al servidor, como la inserción de una imagen.



Interacción entre un cliente y un servidor web

4.4. Parámetros de consulta

Se pueden pasar parámetros de consulta a dentro del URL. Por eso, después del nombre de la página, hay que mencionar un punto de pregunta '?' y uno a varios parámetros deparados por '&', con el nombre del parámetro, el signo '=', y el valor del parámetro.

Veremos ejemplos utilizando los formularios.

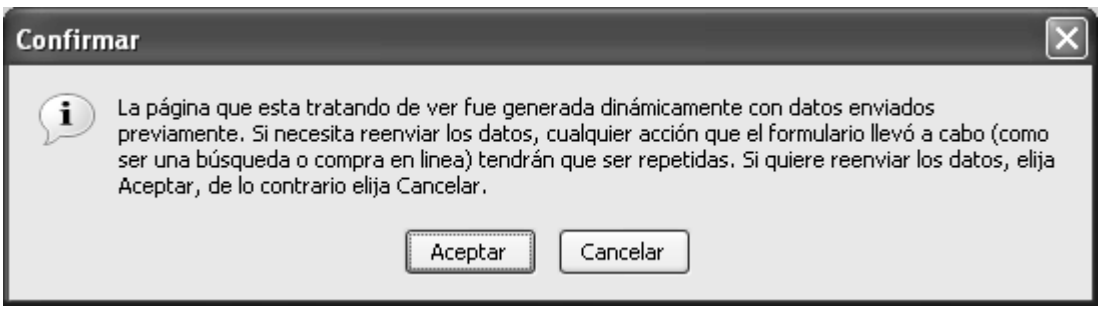
4.5. Métodos de consulta

Ya hemos visto el uso del método GET. También existe el método POST. La diferencia entre los dos es que el método GET pasa los parámetros a dentro del URL, así que se puede guardar fácilmente en su lista de marcadores. También, en navegador sabe que este método no puede dañar el servidor, así acepta de someter la consulta nuevamente (refresca) sin alerta.

El método POST, en vez de pasar los parámetros en el URL, les pasa a dentro del cuerpo de la consulta. Eso permite enviar consulta de tamaño más grande que 2000 caracteres, y sin que los parámetros aparecen en el URL.

Como los parámetros están guardados a dentro del cuerpo de la consulta, la consulta no puede ser guardada tan fácilmente en los marcadores, porque además del URL, deberá guardar el cuerpo de la consulta.

También, cuando se proba de enviar de nuevo una encuesto POST (refresh) el navegador genera una alerto porque este acción puede generar acciones posiblemente irreversible al nivel del servidor.



Ejemplo de formulario:

```
<form action="/prueba.php" method="POST">
Cuidad: <input name="cuidad" type="text">
Pais: <input name="pais" type="text">
<p>
<input type="SUBMIT">
</form>
```

Ejemplo de consulta POST generada por el formulario arriba:

```
POST / prueba.php HTTP/1.0
Host: www.businesssoft.com.ni
User-Agent : Mozilla/4.5 [en] (WinNT; I)
Accept: image/gif, image/jpeg, image/pjpeg, image/png, */*
Accept-language : en
Accept-charset : iso-8859-1,*,utf-8
cuidad=Managua&pais=Nicaragua
```

5. Instalación del PHP

5.1. Objetivo del capitulo

Al fin de este capitulo, el alumno tendrá un servidor PHP y las herramientas para desarollar en PHP instalados en su computadora. Este es un requisito para poder cumplir los ejercicios prácticos.

5.2. Cual versión?

Existen varias versiones de PHP. La útlima versión es la versión 5, y vamos a utilizar esta versión.

5.3. Instalación de PHP

PHP provee un paquete de instalación para Windows. El paquete de instalaciónprobara de configurar tamben el servidor web (IIS/Apache2). Si falla, usted tendrá que confirgurar el servidor web manualmente (documentación incluida en paquete de instalación).

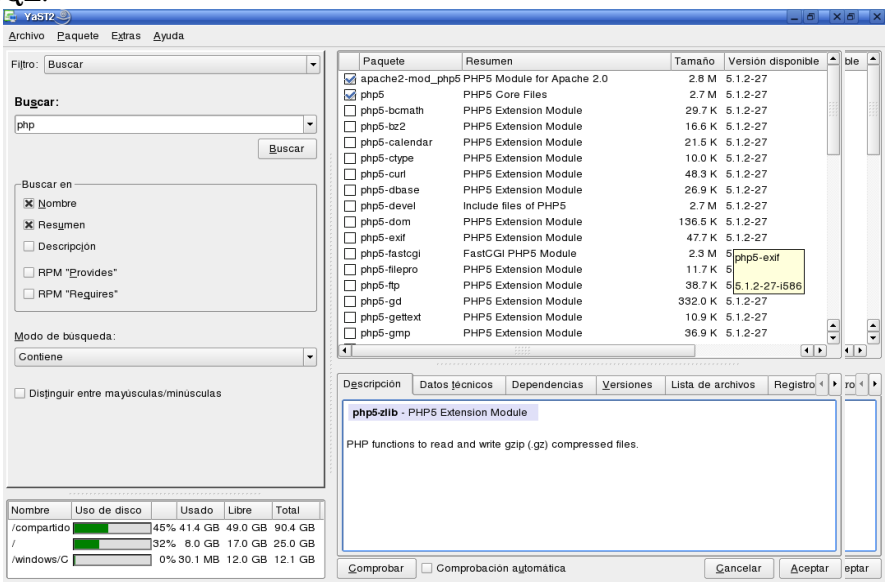
Bajo Linux existen dos posibilidades para instalar PHP: compilar el packete, o utilizar una paquete precompilado (si existe para su versión de Linux).

Sitio web de PHP: http://www.php.net

La versión 10.1 de Suse Linux trae la versión 5 de PHP en sus paquetes disponibles en el CD/DVD.

Hay que notar que PHP no puede funccionar solo, si no que el tiene que funcionar con un servidor Web, como Apache or IIS. En este curso vamos a utilizar PHP 5 con Apache2.

Vamos a instalarlos utilizando YAST, el progarama de instalación y configuración de Linux. Hay que buscar sobre PHP y seleccionar el conector apache2-modphp5, php5, php5-mysqli, php5-mbstring, y php-zlib. Los dos últimos estan utilizados por el sitio web phpMyAdmin, que permite manejar la base de datos MySQL.



5.4. Instalación de Eclipse y del plugin para PHP

Para diseñar las páginas PHP, vamos a utilizar la herramienta Eclipse versión 3.2, con el plugin para PHP.

Estos programas son libres y gratuitos, y funcionan bajo Windows como bajos Linux.

Para instalar Eclipe vamos a abrir una ventana de consola y ponernos como Root (su).

Primero, Eclipse versión 3.2 necesita Java versión 1.5 o arriba. Su el Java 1.5 no está instalado hay que instalarlo primero:
De desde el CD rom:
```
rpm  -iv jdk-1_5_0_07-linux-i586.rpm
```

Ahora el Java esta disponible en en sistema bajo /usr/java/jdk1.5.0_07 pero el Java por defecto es todavia la version 1.4.2 que venia con Suse. Se puede verificar ejecutando `java -version`

Para que la version por defecto sea la version 1.5 hay que cambiar algunos enlaces debajo de `/etc/alternatives`. Para eso, ejecuta desde el CD '. `modifySuseJDK15.sh`'. Si sale un error de que algun archivo no existe, ignorala.

Ahora podemos instalar Eclipse y sus plugins:
Como root, vamos a copiar el archivo de instalación de Eclipse y luego decomprimirlo.
De desde el CD rom:
```
cp eclipse-SDK-3.2-linux-gtk.tar.gz /opt
cd /opt
tar -xvf eclipse-SDK-3.2-linux-gtk.tar.gz
cd /media/PHP
cp net.sourceforge.phpeclipse_1.1.8.bin.dist.zip /opt/eclipse
cd /opt/eclipse
unzip -o net.sourceforge.phpeclipse_1.1.8.bin.dist.zip
cd /opt/eclipse
chgrp users -R *
```


Vamos a crear la carpeta que utilizaremos con Eclipse.
```
md /workspace
chmod 777 -R /workspace/
```

Vamos a copiar el archivo de instalación de Eclipse y luego descomprimirlo.
```
cp Eclipse.desktop /opt/kde3/share/applnk/Development
cp Eclipse.desktop /usr/share/applications
```

Ahora Eclipse esta en el menu de aplicaciones (y de desarrollo).

5.4.1. Inicialización del plugin para PHP

Para poder manejar PHP de desde Eclipse, hay que configurar el plugin.

Debajo del menu 'Window', elegir 'Preferences' y el objeto PHPeclipse Web Development. Seleccionar PHP external tools y corregir el camino hacia PHP. En nuestro caso /usr/bin/php5.

5.4.2. Configuración de Apache

Para que Apache lea las páginas que creamos en Eclipse, tenemos que configurar Apache.

Hay que abrir el archivo `/etc/apache2/httpd.conf` y al final adjuntar las líneas siguientes:

```
NameVirtualHost *:80
########## Apache por defecto ##########
<VirtualHost *:80>
 ServerName localhost
```

```
 ServerAdmin cedric@solucionjava.com
AliasMatch ^/manual(?:/(?:de|en|es|fr|ja|ko|ru))?(/.*)?$ "/usr/share/apache2/manual$1"
<Directory "/usr/share/apache2/manual">
    Options Indexes
    AllowOverride None
    Order allow,deny
    Allow from all
    <Files *.html>
        SetHandler type-map
    </Files>
    SetEnvIf Request_URI ^/manual/(de|en|es|fr|ja|ko|ru)/ prefer-language=$1
    RedirectMatch 301 ^/manual(?:/(de|en|es|fr|ja|ko|ru)){2,}(/.*)?$ /manual/$1$2
</Directory>
</VirtualHost>

########## PHP (Eclipse workspace) ##########
<VirtualHost *:80>
 DocumentRoot /workspace
 ServerName alumnoX (remplaza X por el numero de su maquina)
 ServerAdmin cedric@solucionjava.com
    <Directory "/workspace">
       AllowOverride None
       Order allow,deny
       Allow from all
    </Directory>
</VirtualHost>
```

Ahora tenemos dos sitios web en nuestra máquino: http://localhost que apunta al sitio Apache por de defecto, y http://alumnoX que apuntara a la carpeta de Eclipse (error 403 / acceso prohibido por defecto).

5.5. Instalación de MySQL

En este curso vamos a utilizar la base de datos MySQL, en su versión 5.0.27. Por eso la vamos a instalar ya.

Para instalar MySQL vamos primero a entrar como el usuario Root (o usar su).

Luego abrimos una ventana de consola, introducemos el CD del curso, y vamos a instalar la version de MySQL que estaen el CD lanzando desde el CD la instrucción:

`rpm -iv MySQL-server-5.0.27-0.glibc23.i386.rpm` para instalar el sevidor

`rpm -iv MySQL-client-5.0.27-0.glibc23.i386.rpm` para instalar el cliente

Eso installo MySQL bajo /usr/bin.

Vamos a crear una carpeta /mysql conteniendo los atajos hacia programas de MySQL.

```
. createMySQLlinks.sh
```

Vamos ahora a cambiar la clave del usuario root. Para cambiar la clave, entra en /mysql y ejecuta : `/usr/bin/mysqladmin -u root password 'SolPHP'`. La nueva clave sera 'SolPHP'.

Para verificar que MySQL esta bien instalado y se inicia, ejecuta `'rcmysql restart'` como Root.

Y ahora vamos a crear la base de datos del curso:

```
cd /media/PHP
/mysql/mysql -u root -pSolJava
create database curso;
exit;
/mysql/mysql -u root -pSolJava curso < curso.sql
```

E instalar un entorno de desarrollo (en PHP!) para poder visualizar la base de datos:

```
cp phpMyAdmin.tar.gz /workspace
cd /workspace
tar -xvf phpMyAdmin.tar.gz
```

6. Primera página PHP

6.1. Objetivo del capitulo

Al fin de este capitulo, el alumno hará creado y ejecutado su primer código PHP.

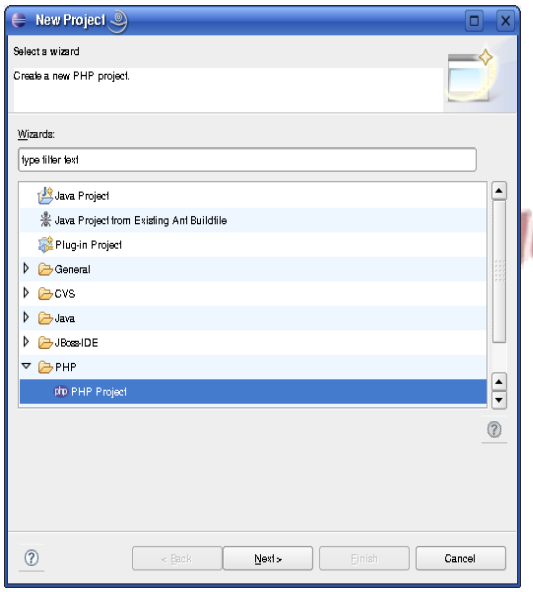
Esta primera página servirá de base para poner en práctica y probar la teoría que vamos a ver en los próximos capítulos.

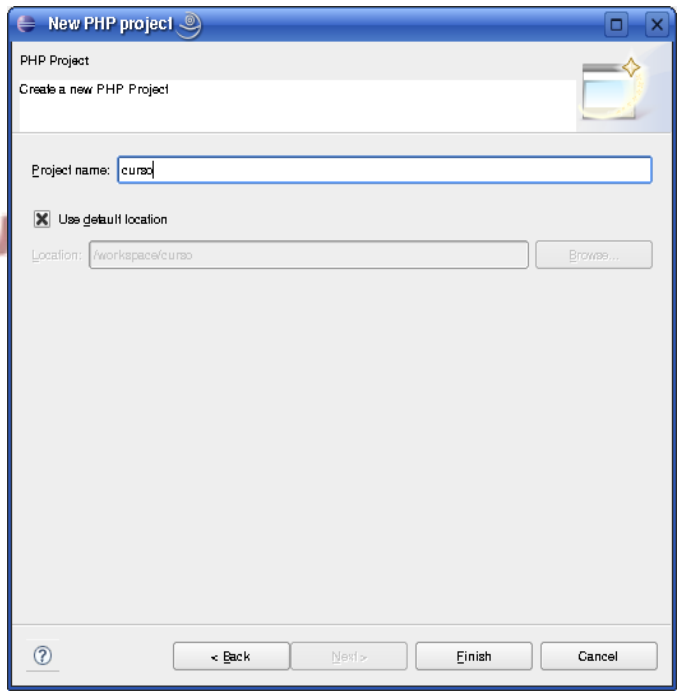
6.2. Creación de un nuevo proyecto web

Al iniciar Eclipse elegimos el workspace '/workspace'.

Para crear nuestra primera página PHP, necesitamos crear un sitio web. Por eso, en Eclipse, creamos un nuevo proyecto PHP, que llamamos ‘curso’, con los valores por defecto.

Al iniciar Eclipse elegimos el workspace '/workspace'.





6.3. Creación de la página de índice

Para poder probar el servidor, vamos a crear nuestra primera página PHP.

Por eso, hacemos un clic derecho sobre la carpeta curso, y elegimos ‘new’...’PHP File’, y en la ventana que se abre, mencionamos en nombre del archivo: index.php.

Todas la páginas con extensión php (.php en minuscula) serán analizadas por el servidor PHP y producirán una página HTML como salida.

Para probar si la página funciona, tenemos que llenarla con algún código PHP y/o HTML. Por defecto la pagina index,php que hemos creado ya esta abierta. Miramos que ya viene con algún código pre-hecho.

Vamos a modificar el código de la manera siguiente, y lo guardamos:

```
<?php
/*
 * Created on 21/07/2006
 *
 * To change the template for this generated file go to
 * Window - Preferences - PHPeclipse - PHP - Code Templates
 */
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<?php
$titulo="Mi primera P&aacute;gina PHP";
$nrPagina=1;
?>
<title><?PHP echo $titulo; ?></title>
</head>
<body bgcolor="#FFFFFF">
<h1><?PHP echo $titulo; ?></h1>
<hr>
<p>
Este es mi p&aacute;gina en PHP numero <?PHP echo $nrPagina; ?>.
</body>
</html>
```

Luego miramos el resultado con un navegador web en el domicilio: http://alumnoX/curso/ o http://alumnoX/curso/index.php

Así podemos ver que en la barra de aplicación como en la primera línea de la página sale en valor de la variable 'titulo'.

1.1

7. Utilización de los scriptlets

7.1. ¿Qué son los scriptlets?

Los scriptlets son las etiquetas que permiten de delimitar el código PHP, que tiene que ser analizado por el servidor PHP, del código HTML, que tiene que ser enviado al cliente como tal. Así el servidor PHP combina el código HTML con el resultado del código PHP para obtener la página HTML que será regresada al cliente y sera leída por el navegador del cliente.

El PHP es la parte de la programación que se ejecuta al nivel del servidor, en contra del `JavaScript`, que se ejecuta al lado del cliente.

Para interpretar un archivo, el servidor PHP símplemente lea el texto del archivo hasta que encuentra uno de los carácteres especiales que delimitan el inicio de código PHP. El intérprete ejecuta entonces todo el código que encuentra, hasta que encuentra una etiqueta de fin de código, que le dice al intérprete que siga ignorando el código siguiente. Este mecanismo permite embeber código PHP dentro de HTML: todo lo que está fuera de las etiquetas PHP se deja tal como está, mientras que el resto se interpreta como código.

Hay cuatro conjuntos de etiquetas que pueden ser usadas para denotar bloques de código PHP. De estas cuatro, sólo 2 (<?php. . .?> y <script language="php">. . .</script>) están siempre disponibles; el resto pueden ser configuradas en el fichero de php.ini para ser o no aceptadas por el intérprete.

Las etiquetas soportadas por PHP son:

1. `<?php echo("si quieres servir documentos XHTML o XML, haz como aqu&iacute;\n"); ?>`
2. ```
   <? echo ("esta es la más simple, una instrucción de procesado SGML \n"); ?>
    <?= expression ?> Esto es una abreviatura de "<? echo expression ?>"
   ```
3. ```
   <script language="php">
        echo ("muchos editores (como FrontPage) no
               aceptan instrucciones de procesado");
    </script>
   ```
4. ```
   <% echo ("Opcionalmente, puedes usar las etiquetas ASP"); %>
    <%= $variable; # Esto es una abreviatura de "<% echo . . ." %>
   ```

El método primero, `<?php. . .?>`, es el más conveniente, ya que permite el uso de PHP en código XML como XHTML, y que esta reconocida por los entornos de desarollo. Es el metodo que vamos a utilizar en el curso.

7.2. Separación de instrucciones

Las separación de instrucciones se hace terminando cada declaración con un punto y coma.
La etiqueta de fin de bloque (?>) implica el fin de la declaración, por lo tanto lo siguiente es equivalente:

```
<?php echo "This is a test"; ?>
<?php echo "This is a test" ?>
```


```

# 8. Características del lenguaje PHP

## 8.1. Objetivo del capitulo

Al fin de este capitulo, el alumno será capaz de entender el uso de variables y los tipo de datos utilizados en PHP.

El podrá también poner comentarios en su código, convertir datos de un tipo a otro, y hacer operaciones sobre variables.

## 8.2. Lista de Palabras Clave

Estas palabras tienen un significado especial en PHP. Algunas de ellas representan cosas que lucen como funciones, o algunas se ven como constantes, y así sucesivamente--pero no lo son, en realidad: son construcciones del lenguaje. Usted no puede usar ninguna de las siguientes palabras como constantes, nombres de clase, nombres de funciones o métodos. Usarlas como nombres de variables está bien, generalmente, pero puede conducir a confusiones.

**Lista de Palabras Clave:**

| | | | | |
|---|---|---|---|---|
| and | or | xor | __FILE__ | exception |
| __LINE__ | array() | as | break | case |
| class | const | continue | declare | default |
| die() | do | echo() | else | elseif |
| empty() | enddeclare | endfor | endforeach | endif |
| endswitch | endwhile | eval() | exit() | extends |
| for | foreach | function | global | if |
| include() | include_once() | isset() | list() | new |
| print() | require() | require_once() | return() | static |
| switch | unset() | use | var | while |
| __FUNCTION__ | __CLASS__ | __METHOD__ | final | php_user_filter |
| interface | implements | extends | public | |
| protected | abstract | clone | try | |
| throw | private | catch | this | |



## 8.3. Sensibilidad

PHP es a veces sensible a las mayúsculas y las minúsculas, a veces no. Así '$miVariable' es una variable diferente de '$MiVariable' pero false o FALSE son iguales.

## 8.4. Identificadores

Los identificadores son los nombres que se dan para identificar a las clases, funciones, variables, constantes o cualquiera etiqueta PHP.

El nombre de un identificador tiene que cumplir con ciertas reglas:

- Debe tener un o mas caracteres
- El primer carácter tiene que ser una letra o el carácter '_' (subrayado).
- Las letras que pueden ser utilizadas después del primer carácter son cualquier número de letras, números y rayas. Como expresión regular se podría expresar como: '[a-zA-Z_\x7f-\xff] [a-zA-Z0-9_\x7f-\xff]*'

## 8.5. Declaración de variables

Las variables permiten almacenar datos de entrada, de salida, o intermedios.

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas.

La sintaxis básica de declaración de variable es:

```
$<nombre de variable> = <valor>;
$miNumero=1;
```

PHP4 y arriba ofrece otra forma de asignar valores a las variables: asignar por referencia. Esto significa que la nueva variable simplemente referencia (en otras palabras, "se convierte en un alias de" ó "apunta a") la variable original. Los cambios a la nueva variable afectan a la original, y viceversa. Esto también significa que no se produce una copia de valores; por tanto, la asignación ocurre más rápidamente. De cualquier forma, cualquier incremento de velocidad se notará sólo en los bucles críticos cuando se asignen grandes matrices u objetos.

Para asignar por referencia, simplemente se antepone un signo "&" al comienzo de la variable cuyo valor se está asignando (la variable fuente). Por ejemplo, el siguiente trozo de código produce la salida 'Mi nombre es Bob' dos veces:

```
<?php
$foo = 'Bob';              // Asigna el valor 'Bob' a $foo
$bar = &$foo;              // Referencia $foo via $bar.
$bar = "Mi nombre es $bar";  // Modifica $bar...
echo $foo;                 // $foo tambien se modifica.
echo $bar;
?>
```


Algo importante a tener en cuenta es que sólo las variables con nombre pueden ser asignadas por referencia.

```
<?php
$foo = 25;
$bar = &$foo;      // Esta es una asignacion valida.
$bar = &(24 * 7);  // Invalida; referencia una expresion sin nombre.

function test() {
   return 25;
}

$bar = &test();    // Invalida.
?>
```

8.6. Tipos simples (o primitivos)

PHP soporta ocho tipos primitivos.

Cuatro tipos escalares:

- boolean
- integer
- float (número de punto-flotante, también conocido como 'double')
- string

Dos tipos compuestos:

- array
- object

Y finalmente dos tipos especiales:

- resource
- NULL

El tipo de una variable usualmente no es declarado por el programador; en cambio, es decidido en tiempo de compilación por PHP dependiendo del contexto en el que es usado la variable.

8.6.1. Boolean

Este es el tipo más simple. Un boolean expresa un valor de verdad. Puede ser TRUE or FALSE.

8.6.1.1. Sintaxis

Para especificar un literal booleano, use alguna de las palabras clave TRUE o FALSE. Ambas son insensibles a mayúsculas y minúsculas.

```
<?php
$foo = True; // asignar el valor TRUE a $foo
?>
```

Usualmente se usa algún tipo de operador que devuelve un valor boolean, y luego éste es pasado a una estructura de control.

```
<?php // == es un operador que prueba por igualdad y devuelve un booleano
if ($accion == "mostrar_version") {echo "La versi&oacute;n es 1.23";}
if ($mostrar_separadores == TRUE) { echo "<hr>\n";} // esto no es necesario...
if ($mostrar_separadores) {echo "<hr>\n";} // ...porque se puede escribir simplemente
?>
```


8.6.1.2. Conversión a booleano

Para convertir explícita mente un valor a boolean, use el moldeamiento (bool) o (boolean). Sin embargo, en la mayoría de casos no es necesario usar el moldeamiento, ya que un valor será convertido automáticamente si un operador, función o estructura de control requiere un argumento tipo boolean.

Cuando se realizan conversiones a boolean, los siguientes valores son considerados FALSE:

- el boolean FALSE mismo
- el integer 0 (cero)
- el float 0.0 (cero)
- el valor string vacío, y el string "0"
- un array con cero elementos
- un object con cero variables miembro (sólo en PHP 4)
- el tipo especial NULL (incluyendo variables no definidas)

Cualquier otro valor es considerado TRUE (incluyendo cualquier resource).

Ejemplo:

```
<?php
var_dump((bool) "");        // bool(false)
var_dump((bool) 1);         // bool(true)
var_dump((bool) -2);        // bool(true)
var_dump((bool) "foo");     // bool(true)
var_dump((bool) 2.3e5);     // bool(true)
var_dump((bool) array(12)); // bool(true)
var_dump((bool) array());   // bool(false)
var_dump((bool) "false");   // bool(true)
```

8.6.2. Enteros

Un integer es un número del conjunto Z = {..., -2, -1, 0, 1, 2, ...}.

8.6.2.1. Sintaxis

Los enteros pueden ser especificados en notación decimal (base-10), hexadecimal (base-16) u octal (base-8), opcionalmente precedidos por un signo (- o +).

Si usa la notación octal, debe preceder el número con un 0 (cero), para usar la notación hexadecimal, preceda el número con 0x.

Ejemplo:

```
<?php
$a = 1234; // numero decimal
$a = -123; // un numero negativo
$a = 0123; // numero octal (equivalente al 83 decimal)
$a = 0x1A; // numero hexadecimal (equivalente al 26 decimal)
?>
```

El tamaño de un entero es dependiente de la plataforma, aunque un valor máximo de aproximadamente dos billones es el valor usual (lo que es un valor de 32 bits con signo).

8.6.2.2. Desbordamiento de enteros

Si especifica un número más allá de los límites del tipo integer, será interpretado en su lugar como un float. Asimismo, si realiza una operación que resulta en un número más allá de los límites del tipo integer, un float es retornado en su lugar.


```
<?php
$numero_grande =  2147483647;
var_dump($numero_grande);
// salida: int(2147483647)

$numero_grande =  2147483648;
var_dump($numero_grande);
// salida: float(2147483648)
?>
```

No hay un operador de división de enteros en PHP. 1/2 produce el float 0.5. Puede moldear el valor a un entero para asegurarse de redondearlo hacia abajo, o puede usar la función round().

```
<?php
var_dump(25/7);         // float(3.5714285714286)
var_dump((int) (25/7)); // int(3)
var_dump(round(25/7));  // float(4)
?>
```

8.6.2.3. Conversión a entero

Para convertir explícitamente un valor a integer, use alguno de los moldeamientos (int) o (integer). Sin embargo, en la mayoría de casos no necesita usar el moldeamiento, ya que un valor será convertido automáticamente si un operador, función o estructura de control requiere un argumento tipo integer. También puede convertir un valor a entero con la función intval().

Desde booleans: FALSE producirá 0 (cero), y TRUE producirá 1 (uno).

Desde números de punto flotante: Cuando se realizan conversiones desde un flotante a un entero, el número será redondeado hacia cero.

Si el flotante se encuentra más allá de los límites del entero (usualmente +/- 2.15e+9 = 2^31), el resultado es indefinido, ya que el flotante no tiene suficiente precisión para dar un resultado entero exacto. No se producirá una advertencia, ¡ni siquiera una noticia en este caso!

Aviso: Nunca moldee una fracción desconocida a integer, ya que esto en ocasiones produce resultados inesperados. `<?php echo (int) ( (0.1+0.7) * 10 ); // imprime 7! ?>`

Desde cadenas: Vea Conversión de cadenas a números

Desde otros tipos: Atención: El comportamiento de convertir desde entero no es definido para otros tipos. Actualmente, el comportamiento es el mismo que si el valor fuera antes convertido a booleano. Sin embargo, no confíe en este comportamiento, ya que puede ser modificado sin aviso.

8.6.3. Números de punto flotante

Los números de punto flotante (también conocidos como "flotantes", "dobles" o "números reales") pueden ser especificados usando cualquiera de las siguientes sintaxis:

```
<?php
$a = 1.234;
$b = 1.2e3;
$c = 7E-10;
?>
```

El tamaño de un flotante depende de la plataforma, aunque un valor común consiste en un máximo de ~1.8e308 con una precisión de aproximadamente 14 dígitos decimales (lo que es un valor de 64 bits en formato IEEE).

8.6.3.1. Precisión del punto flotante

Es bastante común que algunas fracciones decimales simples como 0.1 o 0.7 no puedan ser convertidas a su representación binaria interna sin perder un poco de precisión. Esto puede llevar a resultados confusos: por ejemplo, floor((0.1+0.7)*10) usualmente devolverá 7 en lugar del esperado 8 ya que el resultado de la representación interna es en realidad algo como 7.9999999999....

Esto se encuentra relacionado al hecho de que es imposible expresar de forma exacta algunas fracciones en notación decimal con un número finito de dígitos. Por ejemplo, 1/3 en forma decimal se convierte en 0.3333333. . ..

Así que nunca confíe en resultados de números flotantes hasta el último dígito, y nunca compare números de punto flotante para conocer si son equivalentes. Si realmente necesita una mejor precisión, es buena idea que use las funciones matemáticas de precisión arbitraria o las funciones gmp en su lugar.

8.6.3.2. Conversión a flotante

Para más información sobre cuándo y cómo son convertidas las cadenas a flotantes, vea la sección titulada Conversión de cadenas a números. Para valores de otros tipos, la conversión es la misma que si el valor hubiese sido convertido a entero y luego a flotante. Vea la sección Conversión a entero para más información. A partir de PHP 5, una noticia es generada si intenta convertir un objeto a flotante.

8.6.4. Cadenas

Un valor string es una serie de caracteres. En PHP, un caracter es lo mismo que un byte, es decir, hay exactamente 256 tipos de caracteres diferentes. Esto implica también que PHP no tiene soporte nativo de Unicode.

8.6.4.1. Sintaxis

Un literal de cadena puede especificarse en tres formas diferentes.

- comillas simples
- comillas dobles
- sintaxis heredoc

8.6.4.2. Comillas simples

La forma más simple de especificar una cadena sencilla es rodearla de comillas simples (el caracter ').

Para especificar una comilla sencilla literal, necesita escaparla con una barra invertida (\), como en muchos otros lenguajes. Si una barra invertida necesita aparecer antes de una comilla sencilla o al final de la cadena, necesitará doblarla. Note que si intenta escapar cualquier otro caracter, ¡la barra invertida será impresa también! De modo que, por lo general, no hay necesidad de escapar la barra invertida misma.

Ejemplo:

```
<?php
echo 'esta es una cadena simple';

echo 'Tambi&eacute;n puede tener saltos de l&iacute;nea embebidos
en las cadenas de esta forma, ya que
es v&aacute;lido';

// Imprime: Arnold dijo una vez: "I'll be back"
echo 'Arnold dijo una vez: "I\'ll be back"';

// Imprime: Ha eliminado C:\*.*?
echo 'Ha eliminado C:\\*.*?';

// Imprime: Ha eliminado C:\*.*?
echo 'Ha eliminado C:\*.*?';

// Imprime: Esto no va a expandirse: \n una nueva linea
echo 'Esto no va a expandirse: \n una nueva linea';

// Imprime: Las variables no se $expanden $tampoco
echo 'Las variables no se $expanden $tampoco';
?>
```


8.6.4.3. Comillas dobles

Si la cadena se encuentra rodeada de comillas dobles ("), PHP entiende más secuencias de escape para caracteres especiales (caracteres escapados):

Secuencia	Significado
\n	alimentación de línea (LF o 0x0A (10) en ASCII)
\r	retorno de carro (CR o 0x0D (13) en ASCII)
\t	tabulación horizontal (HT o 0x09 (9) en ASCII)
\\	barra invertida
\$	signo de dólar
\"	comilla-doble
\[0-7]{1,3}	la secuencia de caracteres que coincide con la expresión regular es un caracter en notación octal
\x[0-9A-Fa-f]{1,2}	la secuencia de caracteres que coincide con la expresión regular es un caracter en notación hexadecimal

Nuevamente, si intenta escapar cualquier otro caracter, ¡la barra invertida será impresa también! Antes de PHP 5.1.1, la barra invertida en \{$var} no venía imprimiéndose.

Pero la característica más importante de las cadenas entre comillas dobles es el hecho de que los nombres de variables serán expandidos.

8.6.4.4. Heredoc

Otra forma de delimitar cadenas es mediante el uso de la sintaxis heredoc ("<<<"). Debe indicarse un identificador después de la secuencia <<<, luego la cadena, y luego el mismo identificador para cerrar la cita.

El identificador de cierre debe comenzar en la primera columna de la línea. Asimismo, el identificador usado debe seguir las mismas reglas que cualquier otra etiqueta en PHP: debe contener solo caracteres alfanuméricos y de subrayado, y debe iniciar con un caracter no-dígito o de subrayado.

Aviso
Es muy importante notar que la línea con el identificador de cierre no contenga otros caracteres, excepto quizás por un punto-y-coma (;). Esto quiere decir en especial que el identificador no debe usar sangría, y no debe haber espacios o tabuladores antes o después del punto-y-coma. Es importante también notar que el primer caracter antes del identificador de cierre debe ser un salto de línea, tal y como lo defina su sistema operativo. Esto quiere decir \r en Macintosh, por ejemplo.

Si esta regla es rota y el identificador de cierre no es "limpio", entonces no se considera un identificador de cierre y PHP continuará en busca de uno. Si, en tal caso, no se encuentra un identificador de cierre apropiado, entonces un error del analizador sintáctico resultará con el número de línea apuntando al final del script.

No es permitido usar la sintaxis heredoc al inicializar miembros de clase. Use otro tipo de sintaxis en su lugar.

Ejemplo inválido
```
<?php
class foo {
    public $bar = <<<EOT
bar
EOT;
}
?>
```

El texto heredoc se comporta tal como una cadena entre comillas dobles, sin las comillas dobles. Esto quiere decir que no necesita escapar tales comillas en sus bloques heredoc, pero aun puede usar los códigos de escape listados anteriormente. Las variables son expandidas, aunque debe tenerse el mismo cuidado cuando se expresen variables complejas al interior de un segmento heredoc, al igual que con otras cadenas.

Ejemplo de uso de una cadena heredoc
```
<?php
$cadena = <<<FIN
Ejemplo de una cadena
que se extiende por varias l&iacute;neas
usando la sintaxis heredoc.
FIN;

/* Un ejemplo mas complejo, con variables. */
class foo
{
    var $foo;
    var $bar;
```

```
    function foo()
    {
        $this->foo = 'Foo';
        $this->bar = array('Bar1', 'Bar2', 'Bar3');
    }
}

$foo = new foo();
$nombre = 'MiNombre';

echo <<<FIN
Mi nombre es "$nombre". Estoy imprimiendo algo de $foo->foo.
Ahora, estoy imprimiendo algo de {$foo->bar[1]}.
Esto deber&iacute;a imprimir una letra 'A' may&uacute;scula: \x41
FIN;
?>
```

8.6.4.5. Procesamiento de variables

Cuando una cadena es especificada en comillas dobles o al interior de un bloque heredoc, las variables son interpretadas en su interior.

Existen dos tipos de sintaxis: una simple y una compleja. La sintaxis simple es la más común y conveniente. Esta ofrece una forma de interpretar una variable, un valor array, o una propiedad de un object.

La sintaxis compleja fue introducida en PHP 4, y puede reconocerse por las llaves que rodean la expresión.

➢ ***Sintaxis simple***

Si un signo de dólar ($) es encontrado, el analizador sintáctico tomará ambiciosamente tantos lexemas como le sea posible para formar un nombre de variable válido. Rodee el nombre de la variable de llaves si desea especificar explícitamente el final del nombre.


```
<?php
$cerveza = 'Heineken';
echo "El sabor de varias $cerveza's es excelente"; // funciona, "'" no es un caracter valido para
nombres de variables
echo "Tom&oacute; algunas $cervezas";   // no funciona, 's' es un caracter valido para nombres de
variables
echo "Tom&oacute; algunas ${cerveza}s"; // funciona
echo "Tom&oacute; algunas {$cerveza}s"; // funciona
?>
```

Para cualquier cosa más sofisticada, debería usarse la sintaxis compleja.

➢ ***Sintaxis compleja (llaves)***

Esta no es llamada compleja porque la sintaxis sea compleja, sino porque es posible incluir expresiones complejas de esta forma.

De hecho, de esta forma puede incluir cualquier valor que sea parte del espacio de nombres al interior de cadenas. Simplemente escriba la expresión en la misma forma que lo haría si se encontrara por fuera de una cadena, y luego la ubica entre { y }. Ya que no es posible escapar '{', esta sintaxis será reconocida únicamente cuando el caracter $ se encuentra inmediatamente después de {. (Use "{\$" para obtener una secuencia literal "{$"). Algunos ejemplos para aclarar el asunto:

```
<?php
// Mostremos todos los errores
error_reporting(E_ALL);

$genial = 'fant&aacute;stico';

// No funciona, imprime: Esto es { fant&aacute;stico}
echo "Esto es { $genial}";
```

```
// Funciona, imprime: Esto es fant&aacute;stico
echo "Esto es {$genial}";
echo "Esto es ${genial}";

// Funciona
echo "Esto funciona: {$matriz[4][3]}";

// Funciona. Cuando se usan matrices multi-dimensionales, use siempre
// llaves alrededor de las matrices al interior de cadenas
echo "Esto funciona: {$matriz['foo'][3]}";
?>
```

8.6.4.6. Acceso a cadenas y modificación por caracter

Los caracteres al interior de una cadena pueden ser consultados y modificados al especificar el desplazamiento, comenzando en cero, del caracter deseado después de la cadena entre llaves.

Algunos ejemplos de cadenas

```
<?php
// Obtener el primer caracter de una cadena
$cadena = 'Esta es una prueba.';
$primer = $cadena{0};

// Obtener el tercer caracter de una cadena
$tercer = $cadena{2};

// Obtener el ultimo caracter de una cadena.
$cadena = 'Esta es tambien una prueba.';
$ultimo = $cadena{strlen($cadena)-1};

// Modificar el ultimo caracter de una cadena
$cadena = 'Observe el mar';
$cadena{strlen($cadena)-1} = 'l';
?>
```


8.6.4.7. Funciones y operadores útiles

Las cadenas pueden ser concatenadas usando el operador '.' (punto). Note que el operador '+' (adición) no funciona para este propósito.

Existen bastantes funciones útiles para la modificación de cadenas.

Vea la sección de funciones de cadena en la documentación en el CD del curso para consultar funciones de uso general, o las funciones de expresiones regulares para búsquedas y reemplazos avanzados (en dos sabores: Perl y POSIX extendido).

Existen también funciones para cadenas tipo URL, y funciones para encriptar/descifrar cadenas (mcrypt y mhash).

Finalmente, si aun no ha encontrado lo que busca, vea también las funciones de tipo de caracter.

8.6.4.8. Conversión a cadena

Es posible convertir un valor a una cadena usando el moldeamiento (string), o la función strval(). La conversión a cadena se realiza automáticamente para usted en el contexto de una expresión cuando se necesita una cadena. Esto ocurre cuando usa las funciones echo() o print(), o cuando compara el valor de una variable con una cadena. El contenido de las secciones de la documentación en el CD del curso sobre Tipos y Manipulación de Tipos ayudan a aclarar este hecho.

Un valor boolean TRUE es convertido a la cadena "1", el valor FALSE se representa como "" (una cadena vacía). De esta forma, usted puede convertir de ida y vuelta entre valores booleanos y de cadena.

Un número integer o de punto flotante (float) es convertido a una cadena que representa el número con sus dígitos (incluyendo la parte del exponente para los números de punto flotante).

Las matrices son siempre convertidas a la cadena "Array", de modo que no puede volcar los contenidos de un valor array con echo() o print() para ver lo que se encuentra en su interior. Para ver un elemento, usted tendría que hacer algo como echo $arr['foo']. Vea más adelante algunos consejos sobre el volcado/vista del contenido completo.

Los objetos son convertidos siempre a la cadena "Object". Si quisiera imprimir los valores de variables miembro de un object para efectos de depuración, lea los parágrafos siguientes. Si quiere conocer el nombre de clase del cual un objeto dado es instancia, use get_class(). A partir de PHP 5, el método __toString() es usado si resulta aplicable.

Los recursos son siempre convertidos a cadenas con la estructura "Resource id #1" en donde 1 es el número único del valor resource asignado por PHP durante tiempo de ejecución. Si quisiera obtener el tipo del recurso, use get_resource_type().

NULL se convierte siempre a una cadena vacía.

Como puede apreciar, el imprimir matrices, objetos o recursos no le ofrece información útil sobre los valores mismos. Consulte las funciones print_r() y var_dump() para conocer mejores formas de imprimir valores para depuración.

8.6.4.9. Conversión de cadenas a números

Cuando una cadena es evaluada como un valor numérico, el valor resultante y su tipo son determinados como sigue.

La cadena será evaluada como un float si contiene cualquier caracter entre '.', 'e', o 'E'. De otra forma, evaluará como un entero.

El valor es dado por la porción inicial de la cadena. Si la cadena comienza con datos numéricos válidos, éstos serán el valor usado. De lo contrario, el valor será 0 (cero). Un signo opcional es considerado un dato numérico válido, seguido por uno o más dígitos (que pueden contener un punto decimal), seguidos por un exponente opcional. El exponente es una 'e' o 'E' seguida de uno o más dígitos.

Ejemplo:

```
<?php
$foo = 1 + "10.5";                // $foo es flotante (11.5)
$foo = 1 + "-1.3e3";              // $foo es flotante (-1299)
$foo = 1 + "bob-1.3e3";           // $foo es entero (1)
$foo = 1 + "bob3";                // $foo es entero (1)
$foo = 1 + "10 Cerditos";         // $foo es entero (11)
$foo = 4 + "10.2 Cerditos";       // $foo es flotante (14.2)
$foo = "10.0 cerdos " + 1;        // $foo es flotante (11)
$foo = "10.0 cerdos " + 1.0;      // $foo es flotante (11)
?>
```

8.6.5. Matrices

Una matriz en PHP es en realidad un mapa ordenado. Un mapa es un tipo de datos que asocia valores con claves. Este tipo es optimizado en varias formas, de modo que puede usarlo como una matriz real, o una lista (vector), tabla asociativa (caso particular de implementación de un mapa), diccionario, colección, pila, cola y probablemente más. Ya que puede tener otra matriz PHP como valor, es realmente fácil simular árboles.

Una explicación sobre tales estructuras de datos se encuentra por fuera del propósito de este manual, pero encontrará al menos un ejemplo de cada uno de ellos.

8.6.5.1. Sintaxis

Especificación con array()

Un array puede ser creado por la construcción de lenguaje array(). Ésta toma un cierto número de parejas clave => valor separadas con coma.

```
array( [clave =>] valor
    , ...
    )
// clave puede ser un integer o string
// valor puede ser cualquier valor
```

```
<?php
$matriz = array("foo" => "bar", 12 => true);

echo $matriz["foo"]; // bar
echo $matriz[12];    // 1
?>
```

Una clave puede ser un integer o un string. Si una clave es la representación estándar de un integer, será interpretada como tal (es decir, "8" será interpretado como 8, mientras que "08" será interpretado como "08"). Los valores flotantes en clave serán truncados a valores tipo integer. No existen tipos diferentes para matrices indexadas y asociativas en PHP; sólo existe un tipo de matriz, el cual puede contener índices tipo entero o cadena.

Un valor puede ser de cualquier tipo en PHP.

```
<?php
$matriz = array("unamatriz" => array(6 => 5, 13 => 9, "a" => 42));

echo $matriz["unamatriz"][6];    // 5
echo $matriz["unamatriz"][13];   // 9
echo $matriz["unamatriz"]["a"];  // 42
?>
```

Si no especifica una clave para un valor dado, entonces es usado el máximo de los índices enteros, y la nueva clave será ese valor máximo + 1. Si especifica una clave que ya tiene un valor asignado, ése valor será sobrescrito.

```
<?php
// Esta matriz es la misma que ...
array(5 => 43, 32, 56, "b" => 12);

// ...esta matriz
array(5 => 43, 6 => 32, 7 => 56, "b" => 12);
?>
```

Al usar TRUE como clave, el valor será evaluado al integer 1. Al usar FALSE como clave, el valor será evaluado al integer 0. Al usar NULL como clave, el valor será evaluado a una cadena vacía. El uso de una cadena vacía como clave creará (o reemplazará) una clave con la cadena vacía y su valor; no es lo mismo que usar corchetes vacíos.

No es posible usar matrices u objetos como claves. Al hacerlo se producirá una advertencia: Illegal offset type.

8.6.5.2. Creación/modificación con sintaxis de corchetes cuadrados

Es posible modificar una matriz existente al definir valores explícitamente en ella.

Esto es posible al asignar valores a la matriz al mismo tiempo que se especifica la clave entre corchetes. También es posible omitir la clave, agregar una pareja vacía de corchetes ("[]") al nombre de la variable en ese caso.

```
$matriz[clave] = valor;
$matriz[] = valor;
// clave puede ser un integer o string
// valor puede ser cualquier valor
```

Si $matriz no existe aun, ésta será creada. De modo que esta es también una forma alternativa de especificar una matriz. Para modificar un cierto valor, simplemente asigne un nuevo valor a un elemento especificado con su clave. Si desea remover una pareja clave/valor, necesita eliminarla mediante unset().

```
<?php
$matriz = array(5 => 1, 12 => 2);
$matriz[] = 56;    // Esto es igual que $matriz[13] = 56;
                   // en este punto del script
$matriz["x"] = 42; // Esto agrega un nuevo elemento a la
                   // matriz con la clave "x"
unset($matriz[5]); // Esto elimina el elemento de la matriz
unset($matriz);    // Esto elimina la matriz completa
?>
```

8.6.5.3. Funciones útiles

Existe un buen número de funciones útiles para trabajar con matrices. Consulte la sección funciones de matrices.

Nota: La función unset() le permite remover la definición de claves de una matriz. Tenga en cuenta que la matriz NO es re-indexada. Si sólo usa "índices enteros comunes" (comenzando desde cero, incrementando en uno), puede conseguir el efecto de re-indexación usando array_values().


```
<?php
$a = array(1 => 'uno', 2 => 'dos', 3 => 'tres');
unset($a[2]);
/* producira una matriz que hubiera sido definida como
   $a = array(1 => 'uno', 3 => 'tres');
   y NO
   $a = array(1 => 'uno', 2 =>'tres');
*/

$b = array_values($a);
// Ahora $b es array(0 => 'uno', 1 =>'tres')
?>
```

La estructura de control foreach existe específicamente para las matrices. Ésta provee una manera fácil de recorrer una matriz.

8.6.5.4. Conversión a matriz

Para cualquiera de los tipos: integer, float, string, boolean y resource, si convierte un valor a un array, obtiene una matriz con un elemento (con índice 0), el cual es el valor escalar con el que inició.

Si convierte un object a una matriz, obtiene las propiedades (variables miembro) de ese objeto como los elementos de la matriz. Las claves son los nombres de las variables miembro.

Si convierte un valor NULL a matriz, obtiene una matriz vacía.

8.6.5.5. Ordenamiento de una matriz

Se utiliza la funcion 'sort' sobre el arreglo.

Ejemplo:
```
<?php
sort($archivos);
print_r($archivos);
?>
```

8.6.5.6. Matrices recursivas y multi-dimensionales

Dado que el valor de una matriz puede ser cualquier cosa, también puede ser otra matriz. De esta forma es posible crear matrices recursivas y multi-dimensionales.

Ejemplo:
```
<?php
$frutas = array ( "frutas"  => array ( "a" => "naranja",
                                       "b" => "banano",
                                       "c" => "manzana"
                                     ),
                  "numeros" => array ( 1,
                                       2,
                                       3,
                                       4,
                                       5,
                                       6
                                     ),
                  "hoyos"   => array (      "primero",
                                       5 => "segundo",
                                            "tercero"
                                     )
                );

// Algunos ejemplos que hacen referencia a los valores de la matriz anterior
echo $frutas["hoyos"][5];    // imprime "segundo"
echo $frutas["frutas"]["a"]; // imprime "naranja"
unset($frutas["hoyos"][0]);  // elimina "primero"

// Crear una nueva matriz multi-dimensional
$jugos["manzana"]["verde"] = "bien";
?>
```


8.6.6. Objetos

Existe la posibilidad de utilizar objetos y clases en PHP.

En este curso no vamos ir en detalle sobre la programación orientado objeto porque tomaría demasiado tiempo. Para más detalle, ver la documentación de PHP en el CD del curso.

8.6.7. Recurso

Un recurso es una variable especial, que contiene una referencia a un recurso externo. Los recursos son creados y usados por funciones especiales. Vea la documentación PHP en el CD del curso para un listado de todas estas funciones y los tipos de recurso correspondientes.

8.6.8. NULL

El valor especial NULL representa que una variable no tiene valor. NULL es el único valor posible del tipo NULL.

Una variable es considerada como NULL si
- se ha asignado la constante NULL a la variable.
- no ha sido definida con valor alguno.
- ha sido eliminada con unset().

8.6.8.1. Sintaxis

Existe un solo valor de tipo NULL, y ese es la palabra clave NULL, insensible a mayúsculas y minúsculas.

```
<?php
$var = NULL;
?>
```

8.7. Constantes

Una constante es un identificador para expresar un valor simple. Como el nombre sugiere, este valor no puede variar durante la ejecución del script. (Las constantes especiales __FILE__ y __LINE__ son una excepción a esto, ya que actualmente no lo soin). Una constante es sensible a mayúsculas por defecto. Por convención, los identificadores de constantes suelen declararse en mayúsculas

El alcanze de una constante es global, es decir, es posible acceder a ellas sin preocuparse por el ámbito de alcance.

8.7.1. Sintaxis

Se puede definir una constante usando la función define(). Una vez definida, no puede ser modificada ni eliminada .

Solo se puede definir como constantes valores escalares (boolean, integer, float y string).

Para obtener el valor de una constante solo es necesario especificar su nombre. A diferencia de las variables, no se tiene que especificar el prefijo $. Tambien se puede utilizar la función constant(), para obtener el valor de una constante, en el caso de que queramos expresarla de forma dinámica Usa la función get_defined_constants() parar obtener una lista de todas las constantes definidas.

Si usas una constante todavia no definida, PHP asume que estás refiriéndote al nombre de la constante en si. Se lanzará un aviso si esto sucede. Usa la función defined() para comprobar la existencia de dicha constante.

Estas son las diferencias entre constantes y variables:

- Las constantes no son precedidas por un símbolo de dolar ($)
- Las contantes solo pueden ser definidas usando la función() define , nunca por simple asignación
- Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcanze del ámbito.
- Las constantes no pueden ser redefinidas o eliminadas despues de establecerse; y
- Las constantes solo puede albergar valores escalares

Ejemplo: Definiendo constantes

```
<?php
define("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.
?>
```

8.8. Operadores

Los operadores se utilizan para hacer operaciones sobre las variables. Permiten combinar, modificar, o evaluar las variables.

8.8.1. Principales operadores sobre integer y float

-	Negación (unitario), resta (binario)
++	Incremento
--	Decremento
=	Asignación
==	Igualdad
!=	Desigualdad
<	Menos que
<=	Menos o igual que
>=	Mayor o igual que
>	Mayor que
+	Suma
*	Multiplicación
/	División
%	Modulo

Los operadores de incremento (++) y decremento (--) se pueden utilizar antes o después de la variable, pegado a ella.

Si se utilizan antes, la variable se incrementa o decrementa directamente. Si se utiliza después, la variable se incrementa o decrementa justo después de haber sido utilizada. Se pueden utilizar con cadenas tambien, pero no tiene efecto con boleanos.

Ejemplo de declaración:

```
<?php
echo "<h3>Postincremento</h3>";
$a = 5;
echo "Debe ser 5: " . $a++ . "<br />\n";
echo "Debe ser 6: " . $a . "<br />\n";

echo "<h3>Preincremento</h3>";
$a = 5;
echo "Debe ser 6: " . ++$a . "<br />\n";
echo "Debe ser 6: " . $a . "<br />\n";

echo "<h3>Postdecremento</h3>";
$a = 5;
echo "Debe ser 5: " . $a-- . "<br />\n";
echo "Debe ser 4: " . $a . "<br />\n";

echo "<h3>Predecremento</h3>";
$a = 5;
echo "Debe ser 4: " . --$a . "<br />\n";
echo "Debe ser 4: " . $a . "<br />\n";
?>
```


8.8.2. Operadores de Comparación

Los operadores de comparación, como su nombre indica, le permiten comparar dos valores. Puede que también se encuentre interesado en consultar las tablas de comparación de tipos, ya que éstas muestran ejemplos de varios tipos de comparaciones relacionadas con tipos.

Operadores de Comparación

Ejemplo	Nombre	Resultado
$a == $b	Igual	TRUE si $a es igual a $b.
$a === $b	Idéntico	TRUE si $a es igual a $b, y son del mismo tipo.
$a != $b	Diferente	TRUE si $a no es igual a $b.

Ejemplo	Nombre	Resultado
$a <> $b	Diferente	TRUE si $a no es igual a $b.
$a !== $b	No idénticos	TRUE si $a no es igual a $b, o si no son del mismo tipo.
$a < $b	Menor que	TRUE si $a es escrictamente menor que $b.
$a > $b	Mayor que	TRUE si $a es estrictamente mayor que $b.
$a <= $b	Menor o igual que	TRUE si $a es menor o igual que $b.
$a >= $b	Mayor o igual que	TRUE si $a es mayor o igual que $b.

Si compara un entero con una cadena, la cadena es convertida a un número. Si compara dos cadenas numéricas, ellas son comparadas como enteros. Estas reglas también se aplican a la sentencia switch.

```
<?php
var_dump(0 == "a"); // 0 == 0 -> true
var_dump("1" == "01"); // 1 == 1 -> true

switch ("a") {
case 0:
    echo "0";
    break;
case "a": // nunca se ejecuta ya que "a" ya ha coincidido con 0
    echo "a";
    break;
}
?>
```

Para varios tipos, la comparación se realiza de acuerdo con la siguient tabla (en orden).

Comparación con Varios Tipos

Tipo del Operando 1	Tipo del Operando 2	Resultado
null o string	string	Convertir NULL a "", comparación numérica o de léxico
bool o null	cualquiera	Convertir a bool, FALSE < TRUE
object	object	Las clases internas pueden definir su propia comparación, clases diferentes son incomparables, la misma clase - comparan propiedades en la misma forma que las matrices (PHP 4), PHP 5 tiene su propia explicación
string, resource o number	string, resource o number	Traducir las cadenas y recursos a números, matemática usual
array	array	Una matriz con menos elementos es menor, si una clave del operando 1 no se encuentra en el operando 2 entonces las matrices son incomparables, de otra forma - comparar valor por valor (vea el siguiente ejemplo)
array	cualquiera	array es siempre mayor
object	cualquiera	object es siempre mayor

Ejemplo: Transcripción de la comparación de matrices estándar

```
<?php
// Las matrices son comparadas de esta forma con los operadores de comparacion estandar
function comparacion_matrices_estandar($op1, $op2)
{
    if (count($op1) < count($op2)) {
```

```
        return -1; // $op1 < $op2
    } elseif (count($op1) > count($op2)) {
        return 1; // $op1 > $op2
    }
    foreach ($op1 as $clave => $val) {
        if (!array_key_exists($clave, $op2)) {
            return null; // incomparable
        } elseif ($val < $op2[$clave]) {
            return -1;
        } elseif ($val > $op2[$clave]) {
            return 1;
        }
    }
    return 0; // $op1 == $op2
}
?>
```

8.8.3. Operadores de Lógica

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

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

8.8.4. Operador Ternario

Otro operador condicional es el operador "?:" (o ternario).

Ejemplo:. Asignación de un valor predeterminado

```
<?php
// Ejemplo de uso de: el Operador Ternario
$accion = (empty($_POST['accion'])) ? 'predeterminada' : $_POST['accion'];

// La sentencia anterior es identica a este bloque if/else
if (empty($_POST['accion'])) {
    $accion = 'predeterminada';
} else {
    $accion = $_POST['accion'];
}

?>
```

La expresión (expr1) ? (expr2) : (expr3) evalúa a expr2 si expr1 evalúa a TRUE, y expr3 si expr1 evalúa a FALSE.

Nota: Por favor note que el operador ternario es una sentencia, y que no evalúa a una variable, sino al resultado de una sentencia. Es importante saber esto si se desea devolver una variable por referencia. La sentencia return $var == 42 ? $a : $b; en una función con retorno-por-referencia no funcionará por lo que se ha mencionado y una advertencia es generada en versiones posteriores de PHP.

8.9. Manipulación de Tipos

PHP no requiere (o soporta) la definición explícita de tipos en la declaración de variables; el tipo de una variable es determinado por el contexto en el que la variable es usada. Lo que quiere decir que si asigna un valor de cadena a la variable $var, $var se convierte en una cadena. Si luego asigna un valor entero a $var, ésta se convierte en entera.

Un ejemplo de la conversión automática de tipos de PHP es el operador de adición '+'. Si cualquiera de los operandos es un flotante, entonces todos los operandos son evaluados como flotantes, y el resultado será un flotante. De otro modo, los operandos serán interpretados como enteros, y el resultado será también un entero. Note que este NO modifica los tipos de los operandos mismos; el único cambio está en la forma como los operandos son evaluados.

```
<?php
$foo = "0";  // $foo es una cadena (ASCII 48)
$foo += 2;   // $foo es ahora un entero (2)
$foo = $foo + 1.3;  // $foo es ahora un flotante (3.3)
$foo = 5 + "10 Cerditos"; // $foo es entero (15)
$foo = 5 + "10 Cerdos";   // $foo es entero (15)
?>
```

Si los dos últimos ejemplos lucen extraños, consulte Conversión de cadenas a números.

Nota: El comportamiento de una conversión automática a matriz no se encuentra definido en el momento.

```
<?php
$a = "1";     // $a es una cadena
$a[0] = "f";  // Que hay de las posiciones de cadena? Que sucede?
?>
```


Ya que PHP (por razones históricas) soporta el uso de índices en cadenas mediante desplazamientos de posición usando la misma sintaxis que la indexación de matrices, el ejemplo anterior lleva a un problema: ¿debería $a convertirse en una matriz con un primer elemento "f", o debería "f" convertirse en el primer caracter de la cadena $a?

Las versiones recientes de PHP interpretan la segunda asignación como una identificación de desplazamiento de cadena, así que $a se convierte en "f", sin embargo el resultado de esta conversión automática debe considerarse indefinido. PHP 4 introdujo la nueva sintaxis de llaves para acceder a los caracteres de una cadena, use esta sintaxis en lugar de la que fue presentada anteriormente:

```
<?php
$a    = "abc"; // $a es una cadena
$a{1} = "f";   // $a es ahora "afc"
?>
```

8.9.1. Moldeamiento de Tipos

El moldeamiento de tipos en PHP funciona de forma muy similar a como ocurre en C: el nombre del tipo deseado es escrito entre paréntesis antes de la variable que debe ser moldeada.

```
<?php
$foo = 10;   // $foo es un entero
$bar = (boolean) $foo;   // $bar es un booleano
?>
```

Los moldeamiontos permitidos son:

- (int), (integer) - moldeamiento a entero
- (bool), (boolean) - moldeamiento a booleano
- (float), (double), (real) - moldeamiento a flotante

- (string) - moldeamiento a cadena
- (array) - moldeamiento a matriz
- (object) - moldeamiento a objeto

Note que las tabulaciones y los espacios son permitidos al interior de los paréntesis, así que las siguientes expresiones son funcionalmente equivalentes:

```
<?php
$foo = (int) $bar;
$foo = ( int ) $bar;
?>
```

Nota: En lugar de moldear una variable a cadena, puede también rodear la variable de comillas dobles.

```
<?php
$foo = 10;            // $foo es un entero
$cad = "$foo";        // $cad es una cadena
$fst = (string) $foo; // $fst es tambien una cadena

// Esto imprime "son lo mismo"
if ($fst === $cad) {
    echo "son lo mismo";
}
?>
```

8.9.2. Precedencia de Operadores

La precedencia de un operador indica qué tan "cerca" se agrupan dos expresiones. Por ejemplo, en la expresión *1 + 5 * 3*, la respuesta es *16* y no *18*, ya que el operador de multiplicación ("*") tiene una mayor precedencia que el operador de adición ("+"). Los paréntesis pueden ser usados para marcar la precedencia, si resulta necesario. Por ejemplo: *(1 + 5) * 3* evalúa a *18*. Si la precedencia de los operadores es la misma, se utiliza una asociación de izquierda a derecha.

La siguiente tabla lista la precedencia de los operadores, con aquellos de mayor precedencia listados al comienzo de la tabla. Los operadores en la misma línea tienen la misma precedencia, en cuyo caso su asociatividad decide el orden para evaluarlos.

Precedencia de Operadores:

Asociatividad	**Operadores**	**Información Adicional**
no-asociativo	new	new
izquierda	[	array()
no-asociativos	++ --	incremento/decremento
no-asociativos	! ~ - (int) (float) (string) (array) (object) @	tipos
izquierda	* / %	aritmética
izquierda	+ - .	aritmética, y cadena
izquierda	<< >>	manejo de bits
no-asociativos	< <= > >=	comparación
no-asociativos	== != === !==	comparación
izquierda	&	manejo de bits, y referencias
izquierda	^	manejo de bits
izquierda	|	manejo de bits

Asociatividad	Operadores	Información Adicional
izquierda	&&	lógicos
izquierda	||	lógicos
izquierda	? :	ternario
derecha	= += -= *= /= .= %= &= |= ^= <<= >>=	asignación
izquierda	and	lógicos
izquierda	xor	lógicos
izquierda	or	lógicos
izquierda	,	varios usos

Ejemplo: Asociatividad
```
<?php
$a = 3 * 3 % 5; // (3 * 3) % 5 = 4
$a = true ? 0 : true ? 1 : 2; // (true ? 0 : true) ? 1 : 2 = 2

$a = 1;
$b = 2;
$a = $b += 3; // $a = ($b += 3) -> $a = 5, $b = 5
?>
```

8.10. Comentarios

En el código PHP se pueden meter comentarios.

Los comentarios de una sola línea se sígnala con '//' o con '#'. Los comentarios de una o varias líneas se sígnala con '/*' al inicio, y '*/' al final.

Ejemplos:
```
/* Este
       es un comentario
              de varias líneas */
int miNúmero2; # Este es una muestra de comentario de una línea
int miNúmero = 1; // Este es una muestra de comentario de una línea
```

9. Decisiones y bucles

9.1. Objetivo del capitulo

Al fin de este capitulo, el alumno será capaz de crear esquema de decisiones en su código, así como crear y manejar bucles.

9.2. if ... elseif...else

El `if` se utiliza para crear esquema de decisiones. Se puede utilizar con el `else`, pero es facultativo.

El `if`, que significa 'si' permite probar una o varias condiciones. El resultado de cada condición siempre es si o no. Las condiciones siempre están entre paréntesis.

Después del `if` viene la instrucción que será realizada en caso que el resultado de la(s) condición(es) sale verdadero.

El `else`, que se puede traducir por 'si no se cumplo', la instrucción que será realizada en caso que el resultado de la(s) condición(es) del `if` salió falso.

Si hay mas que una instrucción que ejecutar tras el `if` o el `else`, hay que utilizar un bloque de instrucciones, conteniendo las instrucciones entre { y } .

Existe tambien la instrucción '`elseif`' que significa 'si no se cumplo, si cumple ...' y se trata como un '`if`'.

Ejemplo:

```
<?php
if ($a > $b) {
     print "a es mayor que b";
} elseif ($a == $b) {
     print "a es igual que b";
} else {
     print "a es mayor que b";
}
?>
```

9.3. Sintaxis Alternativa de Estructuras de Control

PHP ofrece una sintaxis altenativa para alguna de sus estructuras de control; a saber, if, while, for, y switch. En cada caso, la forma básica de la sintaxis alternativa es cambiar abrir-llave por dos puntos (:) y cerrar-llave por endif;, endwhile;, endfor;, or endswitch;, respectivamente.

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

En el ejemplo de arriba, el bloque HTML "A es igual 5" se anida dentro de una sentencia if escrita en la sintaxis alternativa. El bloque HTML se mostraría solamente si $a fuera igual a 5.

La sintaxis alternativa se aplica a else y también a elseif. La siguiente es una estructura if con elseif y else en el formato alternativo:

```
<?php
if ($a == 5):
     print "a es igual a 5";
     print "...";
```

```
elseif ($a == 6):
     print "a es igual a 6";
     print "!!!";
else:
     print "a no es ni 5 ni 6";
endif;
?>
```

9.4. switch

El `switch` se utiliza para crear esquema de decisiones. El `switch` permite de tener varias alternativas, en contra del if...else que solo tiene dos alternativas.

Primera se declara, entre paréntesis, cual variable se va a evaluar.

Siguiendo, para cada alternativa, se menciona la palabra `case` con la valor correspondiendo a la valor de la alternativa, y el código que hay que ejecutar. Si no se termina la alternativa con la palabra `break`, el código de las alternativas siguiente se van a ejecutar también, mismo si el valor de la variable evaluada no corresponde al `case`.

Por fin, la palabra `default` se utiliza para mencionar acciones a realizar en caso que ninguna alternativa salió verdadera. Como para el `case`, debería terminar con `break` para que no se ejecutaría otro código siguiendo.

Si hay mas que una instrucción que ejecutar tras el `case` o el `default`, hay que utilizar un bloque de instrucciones, conteniendo las instrucciones entre { y } .

Ejemplo:


```
<?php
$i = 5;
switch ($i) {
     case 0:
         print "i es igual a 0";
         break;
     case 1:
         print "i es igual a 1";
     case 2:
         print "i es igual a 2";
         break;
     default:
         print "i no es igual a 0, 1 o 2";
}
?>
```

9.5. while

El 'while...' se utiliza para crear bucles, es decir repetir una acción por tanto que se cumple a una condición.

La condición que hay que cumplir se menciona detrás de la palabra `while`, entre paréntesis.

Las acciones que hay que ejecutar se mencionan detrás del `while`. Si hay mas que una instrucción que ejecutar, hay que utilizar un bloque de instrucciones, conteniendo las instrucciones entre { y } .

```
<?php
/* ejemplo 1 */

 $i = 1;
 while ($i <= 10) {
     print $i++;  /* el valor impreso ser&iacute;a
                     $i antes del incremento
                     (post-incremento) */
 }
```

```
 /* ejemplo 2 */

 $i = 1;
 while ($i <= 10):
     print $i;
     $i++;
 endwhile;
?>
```

9.6. do ... while

El 'do ... while' se utiliza para crear bucles, es decir repetir una acción por tanto que se cumple a una condición.

La condición que hay que cumplir se menciona detrás de la palabra `while`, entre paréntesis, y termina con punto coma ';'.

Las acciones que hay que ejecutar se mencionan detrás del `do`. Si hay mas que una instrucción que ejecutar, hay que utilizar un bloque de instrucciones, conteniendo las instrucciones entre { y } .

La cláusula `do` se puede mencionar antes de la cláusula `while`. En este caso, el código de la cláusula `do` se ejecutara una primera vez antes de verificar la condición del `while`, y se repetirá hasta que la condición sea falsa.

Ejemplo:

```
<?php
$i=10;
do {
         print "Por aqui paso siempre!";
} while($i<5);
?>
```


9.7. for

El `for` se utiliza para crear bucles un número fijo de veces.

La condición que hay que cumplir se menciona detrás de la palabra `for`, entre paréntesis, y tiene tres diferente partes, separadas por ';'. Cada parte es opcional, pero el ';' tiene que ser mencionado mismo si la parte esta vacía.

La primera parte sirve a declarar un variable (usualmente un int) y/o asignarle un valor original.

La segunda parte contiene la condición de ejecución. Usualmente la variable inicializada > o < que otro int o valor fija.

La tercera parte se utiliza para mencionar el incremento o decremento eventual.

Las acciones que hay que ejecutar se mencionan detrás del `for`. Si hay mas que una instrucción que ejecutar, hay que utilizar un bloque de instrucciones, conteniendo las instrucciones entre { y } .

Ejemplo:

```
<?php
for ($i = 1; $i <= 10; $i++) {
     print $i;
}
?>
```

9.8. foreach

PHP 4 y adelante incluye una construcción `foreach`, tal como perl y algunos otros lenguajes. Esto simplemente da un modo fácil de iterar sobre matrices. `foreach` funciona solamente con matrices y devolverá un error si se intenta utilizar con otro tipo de datos ó variables no inicializadas. Hay dos sintaxis; la segunda es una extensión menor, pero útil de la primera:

```
foreach(expresion_array as $value) sentencia
foreach(expresion_array as $key => $value) sentencia
```

La primera forma recorre el array dado por expresion_array. En cada iteración, el valor del elemento actual se asigna a $value y el puntero interno del array se avanza en una unidad (así en el siguiente paso, se estará mirando el elemento siguiente).

La segunda manera hace lo mismo, salvo que la clave del elemento actual será asignada a la variable $key en cada iteración.

Ejemplo:

```
<?php
$arr = array("one", "two", "three");
foreach ($arr as $value) {
    echo "Value: $value<br>\n";
}

foreach( $arr as $key => $value ) {
   echo "Key: $key; Valor: $value<br>\n";
}
?>
```


9.9. break

La instrucción `break` permite de salir de una bucle o de abandonar las sentencias de ejecución de un `switch`.

Ejemplo:

```
<?php
$ii = 5;
for ($i = 0; $i < 10; $i++) {
        if ($i>$ii) break;
        echo $i;
        }
?>
```

9.10. continue

La instrucción `continue` solo se puede usar en bucles, y permite de saltar directamente a la bucle siguiente, sin ejecutar el resto del código de la bucle corriente.

Ejemplo:

```
<?php
$ii = 5;
for ($i = 0; $i < 10; $i++) {
        if ($i==$ii) continue;
        echo $i;
        }
?>
```

10. Funciones

10.1. Funciones definidas por el usuario

Una función se puede definir con la siguiente sintaxis:

Psuedo código para demostrar el uso de funciones

```
<?php
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Funci&oacute;n de ejemplo.\n";
    return $retval;
}
?>
```

Cualquier instrucción válida de PHP puede aparecer en el cuerpo de la función, incluso otras funiones y definiciones de clases.

10.1.1. Funciones Condicionales

En PHP3, las funciones deben definirse antes de que se referencien. En PHP4 no existe tal requerimiento.Excepto cuando una función es definida condicionalmente como en los ejemplos siguientes.

Cuando una función es definida condicionalmente como se puede ver en estos dos ejemplos, su definición debe ser procesada antes que sea llamada.

Ejemplo:

```
<?php
$makefoo = true;

/* We can't call foo() from here
   since it doesn't exist yet,
   but we can call bar() */

bar();

if ($makefoo) {
  function foo ()
  {
    echo "I don't exist until program execution reaches me.\n";
  }
}

/* Now we can safely call foo()
   since $makefoo evaluated to true */

if ($makefoo) foo();

function bar()
{
  echo "I exist immediately upon program start.\n";
}
?>
```

10.1.2. Funciones dentro de funciones

```
<?php
function foo()
{
  function bar()
  {
    echo "I don't exist until foo() is called.\n";
  }
}
```

```
/* We can't call bar() yet
   since it doesn't exist. */

foo();

/* Now we can call bar(),
   foo()'s processesing has
   made it accessible. */

bar();
?>
```

PHP no soporta la redefinición de funciones previamente declaradas.

Nota: Los nombres de funciones se pueden llamar con mayusculas o minusculas, aunque es una buena costumbre el llamar a las funciones tal y como aparecen en su definición.

10.1.3. Parámetros de las funciones

La información puede suministrarse a las funciones mediante la lista de parámetros, una lista de variables y/o constantes separadas por comas.

PHP soporta pasar parámetros por valor (el comportamiento por defecto), por referencia, y parámetros por defecto. Listas de longitud variable de parámetros sólo están soportadas en PHP4 y posteriores.

10.1.3.1. Pasar parámetros por referencia

Por defecto, los parámetros de una función se pasan por valor (de manera que si cambias el valor del argumento dentro de la función, no se ve modificado fuera de ella). Si deseas permitir a una función modificar sus parámetros, debes pasarlos por referencia.

Si quieres que un parámetro de una función siempre se pase por referencia, puedes anteponer un ampersand (&) al nombre del parámetro en la definición de la función:

Ejemplo pasando parámetros de funciones por referencia

```
<?php
function add_some_extra(&$string)
{
    $string .= ' y algo m&aacute;s.';
}
$str = 'Esto es una cadena, ';
add_some_extra($str);
echo $str;    // Saca 'Esto es una cadena, y algo m&aacute;s.'
?>
```

10.1.3.2. Parámetros por defecto

Una función puede definir valores por defecto para los parámetros escalares estilo C++:

Ejemplo de uso de parámetros por defecto en funciones

```
<?php
function makecoffee ($type = "cappucino")
{
    return "Hacer una taza de $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
?>
```

La salida del fragmento anterior es:

```
Hacer una taza de cappucino.
Hacer una taza de espresso.
```

El valor por defecto tiene que ser una expresión constante, y no una variable, miembro de una clase ó llamada a una función.

Destacar que cuando se usan parámetros por defecto, estos tienen que estar a la derecha de cualquier parámetro sin valor por defecto; de otra manera las cosas no funcionarán de la forma esperada. Considera el siguiente fragmento de código:

Ejemplo de uso incorrecto de parámetros por defecto en funciones
```
<?php
function makeyogurt ($type = "acidophilus", $flavour)
{
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora");   // No funcionar&aacute; de la manera
esperada
?>
```

La salida del ejemplo anterior es:
```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Haciendo un bol de mora.
```

Ejemplo de uso correcto de parámetros por defecto en funciones
```
<?php
function makeyogurt ($flavour, $type = "acidophilus")
{
    return "Haciendo un bol de $type $flavour.\n";
}

echo makeyogurt ("mora");   // funciona como se esperaba
?>
```

La salida de este ejemplo es:
```
Haciendo un bol de acidophilus mora.
```


10.1.3.3. Lista de longitud variable de parámetros

PHP4 soporta las listas de longitud variable de parámetros en las funciones definidas por el usuario. Es realmente fácil, usando las funciones func_num_args(), func_get_arg(), y func_get_args(). Ver la documentación en el CD del curso para más detalles.

No necesita de ninguna sintaxis especial, y las listas de parámetros pueden ser escritas en la llamada a la función y se comportarán de la manera esperada.

10.1.4. Devolviendo valores

Los valores se retornan usando la instrucción opcional return. Puede devolverse cualquier tipo de valor, incluyendo listas y objetos.

Ejemplo de uso de return()
```
<?php
function square ($num)
{
    return $num * $num;
}
echo square (4);   // saca '16'.
?>
```

No puedes devolver múltiples valores desde una función, pero un efecto similar se puede conseguir devolviendo una lista.

Ejemplo retornando una matriz para obtener múltiples valores
```
<?php
function small_numbers()
```

```
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
?>
```

Para retornar una referencia desde una función, se tiene que usar el operador de referencias & tanto en la declaración de la función como en la asignación del valor de retorno a una variable;

Ejemplo retornando una referencia desde una función

```
<?php
function &returns_reference()
{
    return $someref;
}
$newref =& returns_reference();
?>
```

10.2. Funciones internas (incorporadas)

PHP tiene incorporadas muchas funciones y construcciones.

Existen también funciones que requieren extensiones específicas de PHP para que no fallen con un error fatal del tipo "undefined function". Por ejemplo, para usar funciones image, tal como imagecreatetruecolor(), se necesita compilar PHP con soporte para GD. O para usar mysql_connect() se necesita compilar PHP con soporte para MySQL.

Existen muchas funciones en el núcleo de PHP que se incluyen en cada version de PHP, como las funciones string y variable. Una llamada a la función phpinfo() ó get_loaded_extensions() mostrará que extensiones están cargadas en tu versión de PHP. Tener tambien en cuenta que muchas extensiones se encuentran activadas por defecto y que el manual de PHP se encuentra dividido en partes, según estas extensiones. Vea los capítulos configuración, instalación y los capitulos sobre cada extensión en la documentación en el CD del curso, para obtener información sobre como configurar vuestro PHP.

La explicación de como leer e intrerpretar un prototipo de función se encuentra en la sección de la documentación en el CD del curso titulada como leer la definición de una función. Es importante entender que devuelve una función ó si la función trabaja directamente en el valor entregado a la misma. Por ejemplo, str_replace() devuelve una cadena modificada mientras que usort() trabaja directamente en el valor entregado a la misma. Cada página del manual contiene información específica sobre las diferentes funciones existentes, parametros que utilizan, valores devueltos, cambios de comportamiento, etc. Es importante conocer estas diferencias para poder escribir correctamente código PHP.

10.2.1. Unas de las funciones internas más interesantes

10.2.1.1. phpinfo()

```
<?php
   phpinfo();
?>
```

Devuelve la configuración corriente del servidor PHP. Muy intersante para el desarollador. Cuidado en producción que tambien podría ser interesante para hackers...

10.2.1.2. printf

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

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

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

Ejecutar ejemplo Ver código fuente

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

Elemento	Tipo de variable
%s	Cadena de carácteres.
%d	Número sin decimales.
%f	Número con decimales.
%c	Carácter ASCII.

Aunque existen otros tipos, estos son los más importantes.

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

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


10.2.1.3. print_r

print_r() despliega información sobre una variable en una forma que es apta para su lectura por humanos. Si se le entrega una variable tipo string, integer o float, el valor mismo será impreso. Si se le entrega un array, los valores serán presentados en un formato que muestra las claves y los elementos.

Recuerde que print_r() desplazará el apuntador de la matriz al final. Use reset() para llevarlo de vuelta al comienzo.

Ejemplo:

```
<?php
    $a = array ('a' => 'manzana', 'b' => 'banano', 'c' => array ('x', 'y', 'z'));
    print_r ($a);
?>
```

El resultado del ejemplo seria:

```
Array
(
    [a] => manzana
```

```
    [b] => banano
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
```

Si quisiera capturar la salida de print_r(), use el parámetro devolver. Si este parámetro recibe el valor TRUE, print_r() devolverá su salida, en lugar de imprimirla (cosa que hace por defecto).

Ejemplo del parámetro devolver

```
<?php
    $b = array ('m' => 'mono', 'foo' => 'bar', 'x' => array ('x', 'y', 'z'));
    $resultados = print_r($b, true);  // $resultados contiene ahora la salida de print_r
?>
```

10.3. Funciones variables

PHP soporta el concepto de funciones variable, esto significa que si una variable tiene unos paréntesis añadidos al final, PHP buscará una función con el mismo nombre que la evaluación de la variable, e intentará ejecutarla. Entre otras cosas, esto te permite implementar retrollamadas (callbacks), tablas de funciones y demás.

Las funciones varibles no funcionarán con construcciones del lenguaje, tal como `echo(), print(), unset(), isset(), empty(), include(), require()` y derivados. Se necesitará usar una función propia para utilizar cualquiera de estos constructores como funciones variables.

10.3.1. print

`print` -- Muestra una cadena

Descripción: `int print ( string cadena )`

Muestra el valor de cadena por la salida definida. Siempre devuelve el valor 1.

print() no es realmente una función (es una sentencia del lenguaje) de modo que no se requiere el uso de los paréntesis.

10.3.2. echo

`echo` -- Muestra una o más cadenas
Descripción: `void echo ( string arg1 [, string ...] )`

Muestra todos sus parámetros por la salida definida.

echo() no es realmente una función (es una sentencia del lenguaje) de modo que no se requiere el uso de los paréntesis. De hecho, si se indica más de un parámetro, no se pueden incluir los paréntesis.

Ejemplos de echo()

```
<?php
echo "Hola Mundo";
// Se pueden usar variables dentro de una sentencia echo
$saludo = "que tal";
$despedida = "hasta luego";
echo "hola, $saludo"; // hola, que tal
?>
```

10.3.3. require()

La sentencia require() incluye y evalua el archivo especificado.

require() y include() son idénticas en todos los aspectos excepto en el modo de actuar ante un error. include() produce un Warning mientras que require() produce un Error Fatal.

Ver include() para más detalles.

10.3.4. include()

La sentencia include() incluye y evalúa el archivo especificado.

Esta documentación también se aplica a la función require().

require() y include() son idénticas en todos los aspectos excepto en el modo de actuar ante un error. include() produce un Warning mientras que require() produce un Error Fatal. En otras palabras, no dude en utilizar require() si quiere que un fichero no encontrado cuelgue el procesamiento de la página. include() no se comporta de esta manera, el script seguirá funcionando de todas maneras. Asegurarse que include_path este configurado bien.

Cuando un fichero es incluido, el código que contiene hereda la variable scope de la linea en donde el include ocurre. Cualquier variable disponible en esa linea en el fichero desde donde se hace la inclusión estará disponible en el fichero incluido a partir de ese momento.

Ejemplo básico de la funcióninclude()

```
vars.php
<?php
$color = 'green';
$fruit = 'apple';
?>

test.php
<?php
echo "A $color $fruit"; // A
include 'vars.php';
echo "A $color $fruit"; // A green apple
?>
```

Si la inclusión ocurre dentro de una función en el fichero donde se incluye, todo el código contenido en el fichero incluido se comportará como si hubiese sido definido dentro de esta función.

Ejemplo incluyendo desde funciones

```
<?php
function foo()
{
    global $color;
    include 'vars.php';
    echo "A $color $fruit";
}

/* vars.php is in the scope of foo() so     *
* $fruit is NOT available outside of this  *
* scope.  $color is because we declared it *
* as global.                               */
foo();                    // A green apple
echo "A $color $fruit";   // A green
?>
```

Cuando un fichero es incluido, el intérprete sale del modo PHP y entra en modo HTML al principio del archivo referenciado, y vuelve de nuevo al modo PHP al final. Por esta razón, cualquier código

dentro del archivo referenciado que debiera ser ejecutado como código PHP debe ser encerrado dentro de etiquetas válidas de comienzo y fin de PHP.

Ya que include() y require() son contructores especiales del lenguaje, se deben de incluir dentro del bloque de una sentencia, si están dentro de un bloque condicional.

Ejemplo de include() y bloques condicionales

```
<?php
// This is WRONG and will not work as desired.
if ($condition)
    include $file;
else
    include $other;

// This is CORRECT.
if ($condition) {
    include $file;
} else {
    include $other;
}
?>
```

11. Tratamiento de excepciones

Existen tres categorías de errores en PHP. Los errores de compilación, los errores de ejecución, y los errores de lógica. Los errores de ejecución son las que se pueden atrapar en el código y tratar directamente.

11.1. Objetivo del capitulo

Al fin de este capitulo, el alumno será capaz identificar los tipos de errores que se pueden encontrar en un código en PHP, y tratarlos con éxito.

11.2. Errores de compilación

Los errores de compilación son las errores de sintaxis. Al compilar el código, PHP detecta los errores que no respetan la sintaxis del PHP, así un punto-coma faltando al fin de una línea, parentisis abierta y no cerrada, etc...

Estas errores no permiten que se creara el HTML de salida, porque este código no puede ser compilado (analizado) por el servidor PHP. Ningun error está enviada al cliente, pero la página PHP está blanca.

11.3. Errores de lógica

Los errores de lógica son errores debido a un diseño incorrecto del código. Por ejemplo, un bucle que nunca termina, una falta de ortografía en un texto, una formula de cálculo equivocada,...

11.4. Errores de ejecución

PHP 5 tiene un modelo de excepciones similar al de otros lenguajes de programación. Una excepción puede ser lanzada, intentada o capturada en PHP. Un bloque de intento (try) debe incluir por lo menos un bloque de captura (catch). Los bloques de captura múltiples pueden ser usados para capturar diferentes tipos de clases; la ejecución continuará después del último bloque de captura definido. Las excepciones pueden ser lanzadas dentro de bloques de captura.

Cuando es lanzada una excepción, la siguiente línea de código no será ejecutada y PHP intentará encontrar el primer bloque de captura de excepciones. Si una excepción no es capturada se despliega un error fatal de PHP con un mensaje de que la excepción no fue capturada, a menos que exista un manejador de errores definido como set_exception_handler().

11.4.1. Niveles de error de ejecución

Existen varios niveles de error. Se puede definir al nivel del servidor (php.ini) cual(es) nivel(es) de error se van a reportar. En PHP5 por defecto se reporten todas las errores menos las de tipo NOTICE. Un otro parametro indica si el error reportado de ser mostrado (si no, sale una página blanca en vez de la página normal).

Por defecto, los errores están escondidas al usuario (página blanca), por razón de seguridad. Se puede modificar la configuración del servidor (php.ini) para que se enseñan las errores en las páginas a donde ocurren. En producción no esta aconsejado de enseñar errores, que podrían dar información a un eventual 'hacker'.

Errores y Registro

Valor	Constante	Descripción	Nota
1	E_ERROR (integer)	Errores fatales en tiempo de ejecución. Estos indican errores de los que no es posible recuperarse, tales como problemas de asignación de memoria. Se detiene la ejecución del script.	
2	E_WARNING (integer)	Advertencias en tiempo de ejecución (errores no-fatales). La ejecución del script no se interrumpe.	
4	E_PARSE (integer)	Errores de intérprete en tiempo de compilación. Esto tipo de errores deberían ser generados únicamente por el interprete.	
8	E_NOTICE (integer)	Anotaciones en tiempo de ejecución. Indican que el script se ha topado con algo que puede indicar la presencia de un error, pero que también podría ocurrir en el curso normal de la ejecución de un script.	
16	E_CORE_ERROR (integer)	Errores fatales que ocurren durante el arranque inicial de PHP. Es como un E_ERROR, excepto que es generado por el núcleo de PHP.	a partir de PHP 4
32	E_CORE_WARNING (integer)	Advertencias (errores no-fatales) que ocurren durante el arranque inicial de PHP. Es como un E_WARNING, excepto que es generado por el núcleo de PHP.	a partir de PHP 4
64	E_COMPILE_ERROR (integer)	Errores fatales en tiempo de compilación. Es como un E_ERROR, excepto que es generado por el Motor de Scripting de Zend.	a partir de PHP 4
128	E_COMPILE_WARNING (integer)	Advertencias en tiempo de compilación (errores no fatales). Es como un E_WARNING, excepto que es generado por el Motor de Scripting de Zend.	a partir de PHP 4
256	E_USER_ERROR (integer)	Mensaje de error generado por el usuario. Es como un E_ERROR, excepto que es generado desde código PHP usando la función trigger_error().	a partir de PHP 4
512	E_USER_WARNING (integer)	Mensaje de advertencia generado por el usuario. Es como un E_WARNING, excepto que es generado desde código PHP usando la función trigger_error().	a partir de PHP 4
1024	E_USER_NOTICE (integer)	Anotación generada por el usuario. Es como un E_NOTICE, excepto que es generado desde código PHP usando la función trigger_error().	a partir de PHP 4
2047	E_ALL (integer)	Todos los errores y advertencias, en la medida en que sean soportados, excepto por el nivel E_STRICT.	
2048	E_STRICT (integer)	Noticias de tiempo de ejecución. Habilite este valor para hacer que PHP sugiera cambios en su código que velarán por la mejor interoperabilidad y por mantener la compatibilidad de su código.	a partir de PHP 5

Lanzando una Excepción
```
<?php
try {
    $error = 'Always throw this error';
    throw new Exception($error);

    // Code following an exception is not executed.
    echo 'Never executed';

} catch (Exception $e) {
    echo 'Caught exception: ',  $e->getMessage(), "\n";
}

// Continue execution
echo 'Hello World';
?>
```

11.4.2. set_exception_handler

`set_exception_handler` -- Define una función de gestión de excepciones definida por el usuario

11.4.2.1. Descripción:

```
string set_exception_handler ( callback gestor_excepciones )
```

Establece el gestor de excepciones predeterminado si una excepción no es capturada al interior de un bloque try/catch. La ejecución se detendrá después de que gestor_excepciones es llamado.

El gestor_excepciones debe definirse antes de llamar set_exception_handler(). Esta función debe aceptar un parámetro, que será el objeto de excepción que ha sido arrojado.

11.4.2.2. Lista de parámetros

gestor_excepciones: Nombre de la función a ser llamada cuando ocurre una excepción no capturada.

11.4.2.3. Valores retornados

Devuelve el nombre del gestor de excepciones previamente definido, o NULL en caso de error. Si no se había definido un gestor previamente, se devuelve NULL también.

Ejemplo de set_exception_handler()
```
<?php
function gestor_excepciones($excepcion) {
  echo "Excepcion no capturada: " , $excepcion->getMessage(), "\n";
}
set_exception_handler('gestor_excepciones');
throw new Exception('Excepcion No Capturada');
echo "No es ejecutado\n";
?>
```

11.4.2.4. Página de error personalizada

Usted puede mandar a una página de error personalizada utilizando set_exception_handler() para mandar a otro página, por ejemplo una página de error.
En la página de origen:
```
<?php
function gestor_excepciones($num_err, $mens_err, $nombre_archivo, $num_linea, $vars) {
 /* Redireccionar a una pagina diferente en el directorio actual de la peticion */
 $host  = $_SERVER['HTTP_HOST'];
 $uri  = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
 $redirect="error.php?page=".basename($_SERVER['PHP_SELF'])."&error=".$mens_err."&linea=".
$num_linea;
 header("Location: http://$host$uri/$redirect");
 exit;
}
// establecer el gestor de errores definido
$gestor_errores_actual = set_error_handler("gestor_excepciones");
?>
```

error.php

```
<html>
<head>
<meta http-equiv="Content-Language" content="en" />
<meta name="GENERATOR" content="PHPEclipse 1.0" />
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Error</title>
</head>
<body bgcolor="#FFFFFF" text="#000000" link="#FF9966" vlink="#FF9966" alink="#FFCC99">
  <h2>Error en la página :
  <?php
  print $_GET["page"];
  ?> a la linea
  <?php
  print $_GET["linea"];
  ?></h2>
  <hr>
  Error:
  <?php
  $error=  $_GET["error"];;
  print $error;
  ?>
</body>
</html>
```

12. Los formularios

12.1. Creación del formulario

Los formularios están creados simplemente en HTML. El PHP se puede utilizar a dentro del formulario para, por ejemplo, asignar valores por defecto o desactivar ciertos campos según la información disponible (variable de sesión, u otro formulario).

Es muy importante que cada campo del formulario tenga un nombre (parámetro `NAME` del campo).

Para el parámetro `METHOD`, les aconsejo siempre utilizar `POST`, porque este metodo no tiene limite de tamaño de los datos enviados, y esta menos facil de travesiar por el usuario.

12.2. Tratamiento del formulario

Cuando el formulario está enviado, la página PHP a la cual se envío puede recuperar el contenido de cada campo utilizando `$_GET`, `$_POST`, o `$_REQUEST` con en parametro el nombre del parametro de la consulta que queremos recuperar.

`$_GET` solo recupera valores enviadas por el metodo GET, `$_POST` por el metodo POST, y `$_REQUEST` por cualquier metodo.

Si probamos de sacar el valor de un campo que no existe, se genera un error de nivel NOTICE.

Para más facilidad, se pueden guardar los valores en variables.

Ejemplo:
formulario.php (que se podría también llamar formulario.html porque no lleva código PHP)

```
<html>
<head>
<title>Test formulario</title>
</head>
<body bgcolor="#CCDDAA">
<h1>Formulario de prueba</h1>
<hr>
<p>
<form name="formulario" action="resultadoFormulario.php" method=POST>
<table border=0>
<tr>
<th>Su nombre:</th><td><input type=text name="nombre" size=30></td>
<th>Su apellido:</th><td><input type=text name="apellido" size=30></td>
</tr>
<tr>
<th>Su direcciòn:</th><td colspan=3><input type=text name="domicilio" size=100></td>
</tr>
<tr>
<th>Su pais:</th><td><select name="pais">
<option>Bélgica
<option selected>Nicaragua
<option>Otro
</select></td>
</tr>
<tr>
<th>Su no de tel:</th><td><input type=text name="tel" size=30></td>
<th>Su no celucar:</th><td><input type=text name="cel" size=30></td>
</tr>
<tr>
<th>Su nota:</th><td colspan=3><textarea name="nota" cols=75 rows=5></textarea></td>
</tr>
<tr>
<th></th><td><button type=reset>Resetiar</td>
<th></th><td><button type=submit>Enviar</td>
```

```
</tr>
</table>
</form>
</body>
</html>
```

resultadoFormulario.php:

```
<html>
<head>
<?php
$nombre=$_POST["nombre"];
$apellido=$_POST["apellido"];
$omicilio=$_POST["domicilio"];
$pais=$_POST["pais"];
if ($pais==null) $pais="Otro";
$tel=$_POST["tel"];
$cel=$_POST["cel"];
$nota=$_POST["nota"];
?>
<title>Resultado de formulario</title>
</head>
<body bgcolor="#CCDDAA">
<h1>Resultado de su formulario</h1>
<p>
<p>
Estimado/a señor(a) <big><?php print $apellido ?></big>,<br>
<br>
Le/a informamos que su formulario fue transmitido con éxito.
<p>
<?php
if ($pais!="Otro")
{
?>
Notamos que Usted vive en <big><?php print $pais ?></big>, por favor elige su cuidad:<br>
<form name="formulario" action="resultadoFormulario2.php" method=POST>
<select name="cuidad">
<?php
if ($pais=="Nicaragua")
{
?>
<option>Managua
<option>Leon
<option>Granada
<option>Otro
<?php
} // Nicaragua
if ($pais=="Bélgica")
{
?>
<option>Bruselas
<option>Namur
<option>Arlon
<option>Otro
<?php
} // Belgica
?>
</select>
<input type=hidden name="nombre" value="<?php print $nombre ?>">
<input type=hidden name="apellido" value="<?php print $apellido ?>">
<input type=hidden name="domicilio" value="<?php print $domicilio ?>">
<input type=hidden name="pais" value="<?php print $pais ?>">
<input type=hidden name="cel" value="<?php print $cel ?>">
<input type=hidden name="tel" value="<?php print $tel ?>">
<input type=hidden name="nota" value="<?php print $nota ?>">
<button type=submit>Enviar</button>
</form>
<?php
} // <>otro pais
?>
<hr>
<h3>Enlaces</h3>
<a href="index.php">Regresar al indice</a><br>
<a href="formulario.php">Regresar al formulario</a>
</body>
</html>
```

12.2.1. import_request_variables

```
import_request_variables -- Importar variables GET/POST/Cookie en el contexto global
```

12.2.1.1. Descripción

```
bool import_request_variables ( string tipos [, string prefijo] )
```

Importa las variables GET/POST/Cookie en el contexto global. Es útil si usted ha deshabilitado register_globals, pero desea ver algunas variables en el contexto global.

Usando el parámetro tipos, es posible indicar cuáles variables de petición deben importarse. Puede usar los caracteres 'G', 'P' y 'C' respectivamente para indicar GET, POST y Cookie. Estos caracteres no son sensibles a mayúsculas o minúsculas, así que puede usar cualquier combinación de 'g', 'p' y 'c'.

POST incluye la información de archivos cargados mediante POST. Note que el orden de las letras tiene importancia, ya que cuando usa "gp", las variables POST sobrescribirán las variables GET con el mismo nombre. Cualquier otra letra diferente a GPC es descartada.

El parámetro prefijo es usado como prefijo para nombres de variables, que es colocado antes de todos los nombres de variables importados en el contexto global. De modo que si tiene un valor GET llamado "userid", y usa el prefijo "pref_", entonces obtendrá una variable global llamada $pref_userid.

resultadoFormulario2.php:

```
<html>
<head>
<?php
// Esto importara las variables POST con el prefijo "v_"
import_request_variables("P", "v_");
?>
<title>Resultado de formulario</title>
</head>
<body bgcolor="#CCDDAA">
<h1>Resultado final de su formulario</h1>

Su nombre : <?php print $v_nombre ?><br>
Su apellido : <?php print $v_apellido ?><br>
Su domicilio : <?php print $v_domicilio ?><br>
Su cuidad : <?php print $v_cuidad ?><br>
Su pais : <?php print $v_pais ?><br>
Su celular : <?php print $v_cel ?><br>
Su telefono : <?php print $v_tel ?><br>
Su nota : <?php print $v_nota ?><br>
<p>
<hr>
<h3>Enlaces</h3>
<a href="index.php">Regresar al indice</a><br>
<a href="formulario.php">Regresar al formulario</a>
</body>
</html>
```

13. Utilización de COOKIES

13.1. ¿Qué son los COOKIES?

Los COOKIES son variable que se guardan en pequeños archivos de texto en la computadora del cliente y que permiten guardar ciertas informaciones el cliente. Eso permite por ejemplo guardar el nombre del cliente para recuperarlo la próxima vez que el cliente se conecta.

Al contrario de la variable de sesión que se borran al terminar la sesión, se puede definir el tiempo que el COOKIE esta válido. Por defecto, está valido solamente por la sesión corriente (como las variables de sesión), pero se puede cambiar la valor de su tiempo de vencimiento para poder recuperarlo más tarde, en la próxima conexión.

13.2. Creación de un COOKIE

El código de creación de un COOKIE tiene que ir de primero, antes la etiqueta `<HTML>` y de cualquier otro código (PHP,...).

Se utiliza la función `setcookie()` para crear o modificar un cookie.

Los parámetros de `setcookie()` explicados:

Parámetro	Descripción	Ejemplos
nombre	El nombre de la cookie.	'nombre_cookie' es llamada como $_COOKIE['nombre_cookie']
valor	El valor de la cookie. Este valor es almacenado en el equipo del cliente; no almecene información sensible.	Asumiendo que nombre es 'nombre_cookie', este valor es recuperado por medio de $_COOKIE['nombre_cookie']
expirar	La hora en la que expira la cookie. Este valor es una marca de tiempo Unix así que es el número de segundos recorridos desde el epoch. En otras palabras, es probable que este valor sea definido con la función time() más el número de segundos antes de que usted quiera que expire. O es posible usar mktime().	time()+60*60*24*30 definirá que la cookie expire en 30 días. Si no se define, la cookie expirará al final de la sesión (cuando el navegador sea cerrado).
ruta	La ruta en el servidor en la que estará disponible la cookie.	Si se define como '/', la cookie estará disponible en el dominio completo. Si se define como '/foo/', la cookie estará disponible únicamente al interior del directorio /foo/ y todos sus subdirectorios en dominio como /foo/bar/. El valor predeterminado es el directorio actual en el que se define la cookie.
dominio	El dominio en el que la cookie está disponible.	Para lograr que la cookie esté disponible en todos los subdominios de example.com entonces es necesario definir este valor como '.example.com'. El caracter . no es requerido pero hace a la cookie compatible con más

Parámetro	Descripción	Ejemplos
		navegadores. Definir su valor como www.example.com hará que la cookie esté disponible únicamente en el subdominio www. Refiérase a la comparación de sufijos en la especificación para más detalles.
segura	Indica que la cookie debería ser transmitida únicamente sobre una conexión HTTPS segura. Cuando su valor es TRUE, la cookie será definida únicamente si existe una conexión segura. El valor predeterminado es FALSE.	0 o 1

Ejemplo: setCookie.php

```
<?php
$valor = 'algo desde algun lugar';
setcookie("CookieDePrueba", $valor, time()+3600);
setcookie("Prueba", "Hola", time()+3600);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
       <title>Creación de un Cookie</title>
</head>
<body bgcolor="#FFFFFF">
<h1>Cookie creada.</h1>
<hr>
Nombre = CookieDePrueba<br>
Valor = <?php echo $valor ?>
<hr>
<a href="getCookie.php">Leer Cookie</a><br>
</body>
</html>
```


13.3. Recuperación de información de un COOKIE

Una vez se han definido las cookies, ellas pueden ser accesadas en la siguiente carga de página con las matrices $_COOKIE o $HTTP_COOKIE_VARS. Los valores de cookies también existen en $_REQUEST.

Ejemplo: getCookie.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
       <title>Leer variable de sesión</title>
</head>
<body bgcolor="#FFFFFF">
<h1>Lista de Cookies disponibles:</h1>
<hr>
<p>
<?php
print("Imprime una cookie individual : <font color=blue>");
echo $_COOKIE["CookieDePrueba"]."</font><br>";

print("Otra forma de depurar/probar es ver todas las cookies : <font color=blue>");
print_r($_COOKIE);
print("</font><br>");

print("Esto importara las variables Cookie con el prefijo 'v_' : <font color=blue>");
import_request_variables("C", "v_");
print($v_Prueba."</font><br>");
```

```
?>

<table border="1">
<tr><th align=left>Nombre del Cookie</th><th align=left>Valor</th></tr>
<?php
foreach ($_COOKIE as $key => $value)
print("<tr><td>".$key."</td><td><font color=blue>".$value."</td></tr>");
?>
</table>
<hr>
<a href="setCookie.php">Crear un Cookie</a><br>
<a href="invalidCookie.php">Borrar un Cookie</a><br>
</body>
</html>
```

13.4. Borrado de un COOKIE

Para borrar un COOKIE, hay que seguir la misma sintaxis que para crearlo, solo que el valor sea una cadena vacía.

Ejemplo: invalidCookie.php

```
<?php
setcookie("CookieDePrueba", "", time()+3600);
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
       <title></title>
</head>
<body bgcolor="#FFFFFF">
<h1>Su Cookie ha sido borrada.</h1>
<hr>
<a href="getCookie.php">Leer Cookie</a><br>
</body>
</html>
```

14. Utilización de variables de sesión

Las variables de sesión son variables que se pueden atribuir a un cliente conectado, y que serán disponible por el tiempo que dura la sesión: La sesión expira cuando se cierra en navegador, o no se navega en el sitio por un cierto tiempo (depende de la configuración del servidor y se puede modificar en el código PHP), o cuando se invalida la sesión explícitamente en el código PHP.

Cada cliente conectado podrá así tener sus propias valores que se quedaran guardadas en la memoria del servidor.

14.1. Inicio de sesión

En PHP, a menos que el parametro session_autostart sea puesto a 1 en la configuración del servidor, PHP por defecto no abre session por cada usuario conectado. Hay que explicitamente abrir una sesión en el código PHP.

```
<?php
session_start(); // recupera la sesion existente o abre una nueva sesion si no existe ya
?>
```

14.2. Declaración de variable de sesión

Para declara o cambiar el valor de una variable de sesión, asignando un valor a `$_SESSION['miVariable'] = miValor`.

Por ejemplo: `$_SESSION['nombre']=”Cedric”;` crea la variable de sesión nombre con el valor Cedric. Si la variable ya existe, solo le cambia el valor.

14.3. Recuperar el valor de un variable de sesión

Para recuperar el valor de un variable de sesión, se utiliza tambien `$_SESSION['miVariable']`.

Por ejemplo: `$nombre=$_SESSION['nombre'];` regresara el valor de la variable de sesión llamada 'nombre', que en nuestro caso sería Cedric, según el ejemplo de arriba.

Si la variable no existe, regresara un error de nivel NOTICE.

14.4. Invalidar una sesión

Para invalidar una sesión, se utiliza `session_destroy()`.

Por ejemplo: `session.destroy ()` va a invalidar la sesión corriente del usuario.

15. Variables Predefinidas

En PHP hay un sin número de variables que están predefinidas y disponible de desde cualquier parte. Para los detalles de cada variable, ver la documentación disponible en el CD del curso.

15.1. Variables de servidor: $_SERVER

$_SERVER es una matriz que contiene información tal como cabeceras, rutas y ubicaciones de scripts. Las entradas de esta matriz son creadas por el servidor web. No existen garantías de que cada servidor vaya a proveer alguno de estos valores; puede que los servidores omitan algunos, o provean otros que no se listan aquí.

15.2. Variables de entorno: $_ENV

Estas variables son importadas en el espacio de nombres global de PHP desde el entorno bajo el que está siendo ejecutado el intérprete PHP. Muchas son entregadas por el intérprete de comandos bajo el que PHP está corriendo y diferentes sistemas suelen tener diferentes tipos de intérpretes de comandos, una lista definitiva es imposible. Por favor consulte la documentación de su intérprete de comandos por una lista de varibles de entorno que resultan definidas.

15.3. Cookies HTTP: $_COOKIE

Una matriz asociativa de variables pasadas al script actual a través de cookies HTTP. Global automáticamente en cualquier contexto.

15.4. Variables HTTP GET: $_GET

Una matriz asociativa de variables pasadas al script actual a través del método HTTP GET. Global automáticamente en cualquier contexto.

15.5. Variables HTTP POST: $_POST

Una matriz asociativa de variables pasadas al script actual a través del método HTTP POST. Global automáticamente en cualquier contexto.

15.6. Variables de carga de archivos HTTP: $_FILES

Una matriz asociativa de elementos cargados al script actual a través del método HTTP POST. Global automáticamente en cualquier contexto.

15.7. Variables de petición: $_REQUEST

Una matriz asociativa que consiste en los contenidos de $_GET, $_POST, y $_COOKIE.

15.8. Variables de sesión: $_SESSION

Una matriz asociativa que contiene las variables de sesión disponibles en el script actual. Consulte la documentación sobre Funciones de Sesión para más información sobre cómo es usada ésta matriz.

15.9. Variables globales: $GLOBALS

Una matriz asociativa que contiene referencias a todas las variables que están definidas actualmente en el contexto global del script. Los nombres de las variables son las claves de la matriz.

15.10. El mensaje de error previo: $php_errormsg

$php_errormsg es una variable que contiene el texto del último mensaje de error generado por PHP. Esta variable solo estará disponibles dentro del contexto en el que el error ocurrió, y solo si la opción de configuración track_errors está habilitada (por defecto está definida como off).

16. Conexión a MySQL

Cuando se desarrolla una aplicación, muchas veces se necesita conectar a una base de datos. PHP permite conectarse a las mayorías de las base de datos, por tanto que existe un driver ODBC o mejor, un driver PHP.

16.1. Objetivo del capitulo

Al fin de este capitulo, el alumno será capaz de crear una conexión a una base de datos y ejecutar instrucciones en la base de datos. Este capitulo no es una iniciación al SQL ni al manejo de base de datos.

16.2. Driver ODBC

Un driver ODBC (Open Database Connectivity) permite utilizar un driver “genérico” para conectarse a une base de datos. Así PHP se conecta al driver ODBC, y el driver ODBC se conecta a la base de datos. Es el medio más fácil de conectarse a una base de datos, pero es mucho menos eficiente que un driver PHP.

16.3. Driver PHP

PHP provee drivers con funciones predefinidas para la mayoría de las bases de datos.

Para MySQL, existen dos drivers diferentes: `mysql` y `mysqli`. El segundo solo esta disponible en PHP versiones 4.1.3 y arriba, y es una versión mejorada de del driver `mysql`. Permite entre otros llamar a procedimientos, preparar consultas, etc...

En este curso, bien que hemos instalado los dos drivers, solo vamos a utilizar `mysqli`.

Con el fin de ayudar a depurar el programa, cuando falla una consulta SQL o que se proba de conectar, depues de la instrucción se utiliza '`or die ('Mensaje de depuraciñon')`'. En caso de error aparacera el mensaje de depuración en la página PHP del navegador.

16.4. Conexión

Para conectarse a la base de datos MySQL com mysqli, hay que llamar a `mysqli`_connect con 4 parametros: el servidor MySQL, el nombre de usuario, la clave, y la base de datos deseada.

Ejemplo:

```
<?php
$db =mysqli_connect("localhost","root","SolJava","curso") or die('No se pudo conectar a la BD:
' . mysqli_error());
?>
```

16.5. Ejecución de instrucciones SQL

Para poder ejecutar instrucciones SQL en la base de datos, hay que ser conectado a la a base de datos.

Una vez conectado, hay que crear la instrucción, ejecutarla, y eventualmente recoger el resultado.

Dependiendo de si la instrucción debería de regresar datos, o no, se va utilizar una lista de resultados o solamente ejecutar la instrucción.

Ejemplo: mysql.php

```
<?php
$db =mysqli_connect("localhost","root","SolJava","curso") or die('No se pudo conectar a la BD:
' . mysqli_error());
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<?php
$titulo="Conectarse a MySQL";
?>
<title><?PHP echo $titulo; ?></title>
</head>
<body bgcolor="#FFFFFF">
<h1><?PHP echo $titulo; ?></h1>
<hr>
<p>
Lista de paises</p>
<table border="1">
<tr><th>Pais</th></tr>
<?php
$result = mysqli_query($db,"select * from country order by country") or die('consulta 1
fall&oacute;: ' . mysqli_error());
while ($myrow = mysqli_fetch_array($result)) {
print("<tr><td>".$myrow["country"]."</td></tr>");
}
mysqli_free_result($result); // Liberar conjunto de resultados
mysqli_close($db);
?>
</table>
</body>
</html>
```

16.6. consultas preparadas

Para evitar problemas con juegos de carácter o caracteres especiales, es mejor utilizar utilizar consultas preparadas.

Ejemplo de consulta preparada:

```
<?php
$action = $_POST["action"];
$no_country = $_POST["no_country"];
$country = $_POST["country"];
$currency = $_POST["currency"];

$db =mysqli_connect("localhost","root","SolJava","curso") or die('No se pudo conectar a la BD:
' . mysqli_error());
if ($action=="insert" && $country!=null)
{
    /* Preparar la consulta */
    $stmt = mysqli_prepare($db, "insert into country (country, currency) values(?,?)") or
die('consulta 1 fall&oacute;: ' . mysqli_error());

    /* ligar los parametros */
    mysqli_stmt_bind_param($stmt, "ss", $country,$currency);

    /* executar la consulta */
    mysqli_stmt_execute($stmt);

    /* cerrar la consulta */
    mysqli_stmt_close($stmt);
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<?php
$titulo="Conectarse a MySQL";
?>
<title><?PHP echo $titulo; ?></title>
</head>
```

```
<body bgcolor="#FFFFFF">
<h1><?PHP echo $titulo; ?></h1>
<hr>
<form action="mysql2.php" name="form1" method="post">
<p>Lista de paises (limitar al pais numero:
<input type="text" name="no_country" size="4" maxlength="4" onchange="form1.submit();" value="<?
php print $no_country ?>">)
</form></p>
<table border="1">
<tr><th>No</th><th>Pais</th><th>Moneda</th></tr>
<?php
if ($no_country!=null && intval($no_country)>0){
        /* Preparar la consulta */
    $stmt = mysqli_prepare($db, "SELECT country,currency FROM country WHERE no_country=?") or
die('consulta 2 fall&oacute;: ' . mysqli_error());

    /* ligar los parametros */
    mysqli_stmt_bind_param($stmt, "s", $no_country);

    /* executar la consulta */
     mysqli_stmt_execute($stmt);

    /* ligar los campos del resultado */
    mysqli_stmt_bind_result($stmt, $country2, $currency2);

    /* captar el valor */
    mysqli_stmt_fetch($stmt);

    printf("<tr><td>%f</td><td>%s</td><td>%s</td></tr>",$no_country,$country2,$currency2);

    /* cerrar la consulta */
    mysqli_stmt_close($stmt);
}
else {
        $result = mysqli_query($db,"select * from country order by country") or die('consulta 3
fall&oacute;: ' . mysqli_error());
        while ($myrow = mysqli_fetch_array($result)) {
                printf("<tr><td>".$myrow["no_country"]."</td><td>".$myrow["country"]."</td><td>".
$myrow["currency"]."</td></tr>");
        }
        mysqli_free_result($result); // Liberar conjunto de resultados
}
mysqli_close($db);
?>
<form action="mysql2.php" name="form" method="post">
<tr>
<td><input type="submit" value="Adjuntar"></td>
<td><input type="hidden" name="action" value="insert">
<input type="text" name="country" size="10" maxlength="10"></td>
<td><input type="text" name="currency" size="8" maxlength="10"></td>
</tr>
</form>
</table>
</body>
</html>
```


16.7. Llamado a procedimientos

De PHP se pueden también llamar a procedimientos y funciones de la base de datos.

Ejemplo: mysql3.php

```
<?php
$action = $_POST["action"];
$no_country = $_POST["no_country"];

$db =mysqli_connect("localhost","root","SolJava","curso") or die('No se pudo conectar a la BD:
' . mysqli_error());
if ($action=="delete" && intval($no_country)>0)
{
    /* Preparar la consulta */
    $stmt = mysqli_prepare($db, "CALL country_d (?)") or die('consulta 1 fall&oacute;: ' .
mysqli_error());

    /* ligar los parametros */
```

```
    mysqli_stmt_bind_param($stmt, "i", $no_country);

    /* executar la consulta */
    mysqli_stmt_execute($stmt);

    /* cerrar la consulta */
    mysqli_stmt_close($stmt);
}
?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<?php
$titulo="Conectarse a MySQL";
?>
<title><?PHP echo $titulo; ?></title>
</head>
<body bgcolor="#FFFFFF">
<h1><?PHP echo $titulo; ?></h1>
<hr>
<p>Lista de paises</p>
<table border="1">
<tr><th>No</th><th>Pais</th><th>Moneda</th></tr>
<?php
$fromNr=0;
$result = mysqli_query($db,"select * from country order by country") or die('consulta 2
fall&oacute;: ' . mysqli_error());
while ($myrow = mysqli_fetch_array($result)) {
       print("<form action=\"mysql3.php\" name=\"form$fromNr\" method=\"post\">");
       print("<input type=\"hidden\" name=\"action\" value=\"delete\">");
       print("<input type=\"hidden\" name=\"no_country\" value=\"".$myrow["no_country"]."\">");
       print("<tr><td>".$myrow["no_country"]."</td><td>".$myrow["country"]."</td><td>".
$myrow["currency"]."</td>");
       print("<td><input type=\"submit\" value=\"Borrar\"></td>");
       print("</form></tr>");
       $fromNr++;
}
mysqli_free_result($result); // Liberar conjunto de resultados
mysqli_close($db);
?>
</table>
</body>
</html>
```


16.8. Recuperación de fotos en la base de datos

PHP permite recuperar archivos binario de la base de datos y enviarlos al navegador. Un archivo binario puede ser una foto, un archivo de texto, un archivo audio,...

Ejemplo de recuperación de imagen en la base de datos: imagen.php

```
<?php
$db =mysqli_connect("localhost","root","SolJava","curso") or die('No se pudo conectar a la BD:
' . mysql_error());
$id = $_GET["no_photo"];

$sql = "select data, photo_filename, photo_caption, photo_filesize from photo_gallery where
photo_id = $id";
$result= mysqli_query($db,$sql) or die('consulta 1 fall&oacute;: ' . mysqli_error());
while ($myrow = mysqli_fetch_array($result)) {
$data = $myrow["data"];
$photo_filename = $myrow["photo_filename"];
$photo_filetype = $myrow["photo_filetype"];
$photo_caption = $myrow["photo_caption"];
$photo_filesize = $myrow["photo_filesize"];
}
mysqli_free_result($result); // Liberar conjunto de resultados

 header("Content-type: $photo_filetype");
 header("Content-length: $photo_filesize");
 header("Content-Disposition: attachment; filename=$photo_filename");
 header("Content-Description: PHP Generated Data");
  echo $data;
// Cerrar la conexion
```

```
mysqli_close($db);
exit;
?>
```

Ejemplo de página utilizando la foto: index_imagen.php

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<?php
$titulo="Mi primera Página PHP con foto extraida de la BD";
$nrPagina=1;
?>
<title><?PHP echo $titulo; ?></title>
</head>
<body bgcolor="#FFFFFF">
<img src="imagen.php?no_photo=1" align="right">
<h1><?PHP echo $titulo; ?></h1>
<hr>
<p>
Este es mi página en PHP numero <?PHP echo $nrPagina; ?>.
</p>
</body>
</html>
```

17. Autenticación del usuario

17.1. Autentificación HTTP con PHP

PHP provee un mecanismo de autenticación integrado.

Las caracteríticas de autentificación HTTP en PHP solo están disponibles cuando se está ejecutando como un módulo en Apache y hasta ahora no lo estan en la versión CGI. En un script PHP como módulo de Apache, se puede usar la función header() para enviar un mensaje de "Autentificación requerida" al navegador cliente haciendo que muestre una ventana de entrada emergente con nombre de usuario y contraseña.

Ejemplo de autentificación HTTP

```
<?php
  if (!isset($_SERVER['PHP_AUTH_USER'])) {
    header('WWW-Authenticate: Basic realm="My Realm"');
    header('HTTP/1.0 401 Unauthorized');
    echo 'Text to send if user hits Cancel button';
    exit;
  } else {
    echo "<p>Hello {$_SERVER['PHP_AUTH_USER']}.</p>";
    echo "<p>You entered {$_SERVER['PHP_AUTH_PW']} as your password.</p>";
  }
?>
```

Ejemplo de autentificación HTTP forzando una reentrada

```
<?php
  function authenticate() {
    header('WWW-Authenticate: Basic realm="Test Authentication System"');
    header('HTTP/1.0 401 Unauthorized');
    echo "You must enter a valid login ID and password to access this resource\n";
    exit;
  }

  if (!isset($_SERVER['PHP_AUTH_USER']) ||
      ($_POST['SeenBefore'] == 1 && $_POST['OldAuth'] == $_SERVER['PHP_AUTH_USER'])) {
   authenticate();
  }
  else {
   echo "<p>Welcome: {$_SERVER['PHP_AUTH_USER']}<br>";
   echo "Old: {$_REQUEST['OldAuth']}";
   echo "<form action='{$_SERVER['PHP_SELF']}' METHOD='POST'>\n";
   echo "<input type='hidden' name='SeenBefore' value='1'>\n";
   echo "<input type='hidden' name='OldAuth' value='{$_SERVER['PHP_AUTH_USER']}'>\n";
   echo "<input type='submit' value='Re Authenticate'>\n";
   echo "</form></p>\n";
  }
?>
```


17.1.1. Control de acceso a los recursos web

PHP no provee control de acceso integrado. El control de acceso se tiene que programar en el código PHP.

17.1.2. Verificación de la información del usuario

PHP no provee mecanismo predefinido para verificar la información del usuario. La verificación se tiene que programar en el código PHP.

17.1.3. Recuperación de la información del usuario

Una vez que el usuario ha rellenado el nombre y la contraseña, la URL que contiene el script PHP será llamada de nuevo con las variables predefinidas PHP_AUTH_USER, PHP_AUTH_PW, y AUTH_TYPE asignadas con el nombre de usuario, la contraseña y el tipo de autentificación respectivamente. Estas variables predefinidas se pueden encontrar en las matrices $_SERVER y $HTTP_SERVER_VARS. Sólo autentificación "Básica" está soportada en este momento.

17.2. Autenticación manejada por la aplicación

PHP solo provee una autenticación muy pobre, con la cual hay todavía que escribir la mayoría del código.

Por esta razón, se puede utilizar un mecanismo de seguridad manejado completamente por la aplicación, que será, por ejemplo, ligado a una base de datos de usuarios y utilizara un formulario de registro mas simpático.

La implementación de un mecanismo de autenticación de usuario y control de recursos necesita lo siguiente:

1. Registro de usuario
2. Página de autenticación
3. Mecanismo de autenticación, llamado por la página de autenticación
4. Información del usuario guardada al nivel de la sesión, como prueba de que el usuario está autenticado
5. Verificación de la validez de la información de sesión en cada página con acceso restringido.

Al momento que se verifica su nombre de usuario y su nombre, se le asigna una o varias variables de sesión. En la páginas protegidas, se verifica si la variable de sesión existe para este usuario. Si es no existe, es que el usuario probo de llagar a la página con un URL directo, sin autenticarse.

También, si el nombre del usuario es una de las variables de sesión, se puede recuperar de desde cualquiera página PHP después de la autenticación.

Ejemplo de pagina de entrada: login.php

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Login</title>
<script language="javascript">
if (top.frames.length!=0)
top.location=self.document.location;
</script>
</head>
<body onload="document.form.login.focus();" style="background:#77aaff;">
<script type="text/javascript" src="scripts.js"></script>
<script type="text/javascript">
function validate_form(myform)
{
var check_result = "OK";
  clearall();
  if (isNull(myform.login,"Su nombre de usuario","ES"))
  {
  check_result = "NOK";
  }
  if (isNull(myform.password,"Su clave","ES"))
  {
  check_result = "NOK";
  }
  if (check_result == "NOK")
  return(false);
  else
```

```
  return(true);
}
function clearall()
{
         document.form.login.style.backgroundColor ="#CBEDE3";
         document.form.password.style.backgroundColor ="#CBEDE3";
}
</script>
<table width="100%" height="100%" border="0"><tr><td>
<p align="center">
<br><u><b><big>LOGIN</big></b></u>
<br><br>
</p>
<form name="form" action="check_login.php" method="POST" onsubmit="return
(validate_form(form));">
<table border="1" align="center">
<th>Su nombre de usuario</th>
<td><input name="login" type="text" size="30" maxlength="30" Autocomplete="off"></td></tr>
<th>Su clave</th>
<td><input name="password" type="password" size="30" maxlength="30"
  onfocus="this.value='';" Autocomplete="off"></td></tr>
</table><p align="center">
<input type="submit" class=noprint name="submitbt" value="Entrar">   
<input name="userAgent" type="hidden"> <script
type="text/javascript">form.userAgent.value=(navigator.userAgent);</script>
<input name="screen" type="hidden"> <script
type="text/javascript">form.screen.value=screen.width+" x "+screen.height;</script>
</form>
<p align="center">
<script language="javascript" type="text/javascript" >
       if (screen.width<1000) {
       document.writeln("Mejor visto con resoluci&oacute;n de pantalla de 1024 x 768 o arriba.
");
       document.writeln("<i>Su resoluci&oacute;n de pantalla es de "+screen.width+" x
"+screen.height+".</i><br>");
       }
</script></p>
<h2 align="center">
<font color="Blue">
<script language="javascript" type="text/javascript" >
document.writeln("</font><font color=#77aaff>");
</script>
&iexcl;Usted necesita de tener el <big><font color="Red">
<script language="javascript" type="text/javascript" >
document.writeln("</font><font color=#77aaff>");
</script>
Javascript</font></big> encendido<br>en las opciones de su navegador<br>para entrar y utilizar
esta aplicacion web!</font><br>
<font color="Blue">
<script language="javascript" type="text/javascript" >
document.writeln("</font><font color=#77aaff>");
</script>Para m&aacute;s informaci&oacute;n pregunta a su administrador de sistema.</h2>
</td></tr>
<tr><td colspan="2" align="center" style="background-color:#3366CC;">
<a href="mailto:cedric@olucionjava.com" Style="color : White;" title="Este obra esta protegida
por la ley de propiedad intelectual. Para mas informacion, clic aqui para enviar un email a autor
de esta obra."><font size="1">
&copy; Copyright Ing. Cedric Simon, 2006</font></a></font></td>
</tr>
</table>
</body>
</html>
```


Ejemplo de verificación de usuario: check_login.php

```
<?php
session_start();
$login = $_POST["login"];
$password = $_POST["password"];
$noUser=-2;
$db =mysqli_connect("localhost","root","SolJava","curso") or die('No se pudo conectar a la BD:
' . mysql_error());
$result = mysqli_query($db,"select * from usuario where upper(usuario)=upper('".$login."')".
                  " AND DECODE(clave,'SolJava')='".$password."' AND activo=1") or
die('consulta 1 fallo: ' . mysql_error());
if ($myrow = mysqli_fetch_array($result)) {
$noUser=$myrow["no_usuario"];
```

```
$_SESSION['noUser']=$noUser;
$uname=$myrow["fname"]." ".$myrow["lname"];
$_SESSION['uname']=$uname;
$result = mysqli_query($db,"update usuario set bad_try=0 where no_usuario=".$noUser) or
die('consulta 2 fallo: ' . mysql_error());
$redirect="index_secure.php";
// Liberar conjunto de resultados
mysqli_free_result($result);
} else {
$result = mysqli_query($db,"update usuario set bad_try=bad_try+1 where upper(usuario)=upper('".
$login."')") or die('consulta 3 fallo: ' . mysql_error());
$redirect="login.php?nok=badpassword";
}
// Cerrar la conexion
mysqli_close($db);
/* Redireccionar a una pagina diferente en el directorio actual de la peticion */
 $host  = $_SERVER['HTTP_HOST'];
 $uri  = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
 header("Location: http://$host$uri/$redirect");
 exit;
?>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>title</title>
</head>
<body>
</body>
</html>
```

Ejemplo de página segura: index_secure.php

```
<?php require("cabecera_segura.php"); ?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html>
<head>
<?php
$titulo="Mi primera Página PHP segura";
$nrPagina=1;
?>
<title><?PHP echo $titulo; ?></title>
</head>
<body bgcolor="#FFFFFF">
<h1><?PHP echo $titulo; ?></h1>
<hr>
<p>
Este es mi página en PHP segura y usted esta identificado como <font color=blue><?PHP echo
$uname; ?></font>.
</p>
</body>
</html>
```


y cabecera_segura.php

```
<?php
session_start();
if (!session_is_registered('uname')) {
/* Redireccionar a una pagina diferente en el directorio actual de la peticion */
 $host  = $_SERVER['HTTP_HOST'];
 $uri  = rtrim(dirname($_SERVER['PHP_SELF']), '/\\');
 $redirect="login.php";
 header("Location: http://$host$uri/$redirect");
 exit;
} else {
$noUser=$_SESSION['noUser'];
$uname=$_SESSION['uname'];
//$db =mysqli_connect("localhost","root","SolJava","nicacert") or die('No se pudo conectar a la
BD: ' . mysqli_error());
$db =mysql_connect("localhost","root","SolJava") or die('No se pudo conectar a la BD: ' .
mysql_error());
mysql_select_db("relihco_nicacert",$db) or die('No se pudo conectar (2) a la BD: ' .
mysql_error());
}
?>
```

18. Ejercicios

Estos ejercicios le permitirán practicar la materia aprendida en este curso. Las respuestas de los ejercicios se encuentran en el CD del curso (archivo EjerciciosPHP.zip). Existen varias posibilidades de llegar al resultado deseado. En las respuestas en el CD encontraras una o varias posibilidades. Las respuestas en el CD no son exhaustivas...

El objetivo es que logras hacer todos los ejercicios sin ayuda (ni del manual, ni de la documentación, ni de otro código existente). El objetivo no es de copiar y pegar un código para terminar el ejercicio si no de confirmar que entendiste bien e integraste la materia.

Estos ejercicios son una buena preparación para el examen, pero no son suficientes. Tienes que repasar la teoría y practicar mas para prepararte bien al examen de PHP.

Nota: los ejercicios en el CD se conectan a la base de datos 'curso' en 'localhost', con el usuario 'curso' y la clave 'SolJava'.

1. Declara un entero, incrementarlo, decrementarlo, hacer operaciones con el.
2. Probar la prioridad de operadores
3. Probar la conversión entre tipos simples.
4. Declarar una constante e imprimir su valor. Probar de cambiar su valor luego y ver que es lo que pasa.
5. Adjuntar comentario en un su código de tres maneras diferentes.
6. Crear un arreglo conteniendo los números de 1 a 10, e imprimir los números 5 y 10 utilizando el arreglo.
7. Crear un arreglo con nombre, apellido, y fecha de nacimiento de 3 empleados. Imprimir el nombre y la fecha de nacimiento del tercer empleado.
8. Declarar une entero y asignarle un valor. Si el valor < 5 imprimir "Pequeño", si no imprimir "Grande".
9. Declarar une entero y asignarle un valor. Si el valor = 1 imprimir "Uno", = 2 imprimir "Dos", =3 imprimir "3", si no imprimir "Ni uno, ni dos, ni tres", utilizando if...elseif...else...
10. Declarar une entero y asignarle un valor. Si el valor = 1 imprimir "Uno", = 2 imprimir "Dos", =3 imprimir "3", si no imprimir "Ni uno, ni dos, ni tres", utilizando switch.
11. Declarar un entero = 0. Mientras el entero <5, imprimir su valor y incrementarlo de uno, utilizando while.
12. Declarar un entero = 10. Mientras el entero >5, imprimir su valor y decrementarlo de uno, pero imprime por lo menos una vez su valor, utilizando do...while.
13. Crear una bucle que se ejecuta 10 veces utilizando un entero que va de 10 a 19 incrementándose. Imprimir en cada bucle el valor del entero, menos cuando es igual a 15. Utilizando for.
14. Crear una página JSP del los cuales el color del fondo está basado el las segundas de la hora del sistema al momento de la apertura de la página: si la hora tiene entre 0 y 14 segundos, fondo blanco, 15 y 29, color amarillo, 30 y 44, azul claro, y 45 y 59, verde (sin refresco automático).
15. Incluir un archivo texto o HTML externo en una página PHP.
16. Crear una función de adjunta 10 al número pasado en parametro y devuelve el resultado.
17. Crear una función que devuelve los minutos de la hora corriente mas 10 minutos.
18. Crear una función pasandole un parámetro por referencia. Cambiar el valor del parámetro en la función pero no devolver nada. Verificar que la variable referenciada ha sido cambiada también.
19. Crear un formulario de inscripción a un concurso (nombre, apellido, email), y después de haber tratado su contenido, enseñar en otra página el contenido del formulario llenado, en formato de tabla.
20. Adjuntar a la pagina PHP tratando el formulario hecho en el punto anterior unas verificaciones al nivel del servidor (los tres campos no pueden ser vacío, el email debe contener un ‘@’), y en caso

de error (campo vacìo, etc...) regresar al formulario original enseñando los campos con errores y recuperando los valores entradas por el usuario.
21. Crear un formulario de entrada al sitio, y recuperar el nombre de usuario en una variable de sesión y mencionar su nombre en otras páginas relacionadas del mismo sitio.
22. Crear una página para salir (destruir) de su sesión
23. Utilizar un cookie para recordar el nombre de usuario, y proponer el ultimo nombre de usuario por defecto el la página de registro.
24. Crear la opción de borrar el cookie y invalidar la sesión al mismo tiempo
25. Crear una página que enseña todos las variables predefinidas de tipo $_SERVER (clave y valor de cada una). Lo mismo para las otras variables predefinidas ($_SESSION, $_GLOBALS, $_POST...)
26. Crear y probar una página de error personalizada.
27. Crear una pagina que permite ver la lista de empleados y filtrarlos por pais (job_country).
28. Crear una página que lista los empleados (tabla employee), y permite crear, modificar, o borrar empleados.
29. Crear una pagina web conteniendo imágenes que vienen de una base de datos.
30. Crear sitio utilizando los métodos de autenticación HTTP de PHP (usuarios y claves en archivo texto).

Ejercicio final (no se encuentra en el CD del curso):

Crear una aplicación web que:

1) Trae todo su contenido de una base de datos: Textos, etiquetas, imágenes,...
2) Tiene tres partes:
   i. una publica: bienvenida, informaciones generales, pagina de registro, lista de producto sin precio.
   ii. una con autenticación por la aplicación (usuarios en la base de datos): lista de productos con precio, manejo de usuarios y posibilidad de compra en línea.
   iii. Utilizar pagina de error personalizada en todas las paginas
3) Una vez autenticado, el nombre del usuario debe aparecer en todas las pantallas.
4) Manejo de un sistema de compra en línea: el usuario puede llenar una lista de compras, ver las compras anteriores que el hizo...

19. Esquema de la base de datos