Introducción:

Funcionamiento de un Web Site:

El funcionamiento de un Web-Site es un ejemplo típico de la arquitectura cliente-servidor, en donde múltiples clientes se conectan a un servidor (en algunos casos varios) en forma simultanea. En general el servidor depende de la instalación del site mientras que el cliente suele ser un browser, en general Netscape Navigator o Microsoft Explorer. Como en todo esquema cliente-servidor debe existir un protocolo que especifique de que forma se comunican e intercambian datos el cliente y el servidor, el protocolo utilizado en un web site es el protocolo HTTP que funciona "encapsulado" sobre el protocolo TCP/IP.

Introducción al Protocolo HTTP:

Básicamente el protocolo es iniciado por el cliente con un "request", es decir un pedido de un recurso determinado, que es casi siempre contestado por el server con el envío de una respuesta ("response") que incluye un código indicando si el pedido pudo ser resuelto por el server o no.

Un request genérico tiene la forma:

METODO URI PROTOCOLO CrLf
HEADERS*
CrLf
Datos

El MÉTODO en general puede ser GET o POST
URI es el identificador del recurso que se desea pedir, el formato es: http://host:port/path?query_string
PROTOCOLO debe ser HTTP / 1.1
CrLf es un Carriage Return seguido de un New Line (0x13,0x10)
Headers son de tipo: Header-Name: Value CrLf, y pueden indicar varias cosas.

Un ejemplo de pedido es:
GET http://www.yahoo.com HTTP/1.1

El server responde con una RESPUESTA de la forma:

PROTOCOLO STATUS VALOR CrLF
Headers*
Content-Type: TIPO CrLf
CrLf
Datos

Un ejemplo de respuesta de un server podría ser:

HTTP/1.1 200 OK
Date: Mon, 12 Jun 2000 14:04:28 GMT
Server: Apache/1.3.9 (Unix) ApacheJServ/1.1
Connection: close
Content-Type: text/html

Datos…..

Los datos que el server envía al browser dependen del “Content-Type” declarado, básicamente los tipos más usados son texto plano (text/plain), código html (text/html), o imágenes (image/gif u otros).

De esta forma el cliente y el server se comunican por medio de tantos ciclos REQUEST-RESPONSE como sean necesarios, es de destacar que por cada REQUEST se debe iniciar una conexión nueva entre el cliente y el servidor ya que el protocolo HTTP en su forma básica no admite que en una misma conexión se haga más de un pedido al server. En una página html simple con 3 imágenes por ejemplo es normal que se efectúen 4 conexiones al server: una para la página y luego una por cada imagen.

Ejemplo:

Supongamos que tenemos la siguiente página html en un servidor, supongamos que la dirección del servidor es www.prueba.com y que la página se llama index.html, el ciclo que se da entre el browser y el server es de La forma:

```
<HTML>
<HEAD>
<TITLE>Ejemplo</TITLE>
</HEAD>
<BODY>
Hola esta es una prueba
<IMG SRC=”prueba.gif”>
</BODY>
</HTML>
```

Comunicación Browser-Server

BROWSER	**SERVER**
GET http://www.prueba.com/index.html HTTP / 1.1	HTTP/1.1 200 OK Date: Tue, 13 Jun 2000 14:15:45 GMT Server: Apache/1.3.9 (Unix) PHP/4.0.0 Last-Modified: Tue, 13 Jun 2000 14:09:05 GMT ETag: "5804d-73-39464081" Accept-Ranges: bytes Content-Length: 115 Connection: close Content-Type: text/html <HTML> <HEAD> <TITLE>Ejemplo</TITLE> </HEAD> <BODY> Hola esta es una prueba <IMG SRC="prueba.gif"> </BODY> </HTML>
GET http://www.prueba.com/prueba.gif HTTP / 1.1	HTTP/1.1 200 OK Date: Tue, 13 Jun 2000 14:18:22 GMT Server: Apache/1.3.9 (Unix) PHP/4.0.0 Last-Modified: Tue, 13 Jun 2000 14:07:36 GMT ETag: "5804e-2b2-39464028" Accept-Ranges: bytes Content-Length: 690 Connection: close Content-Type: image/gif GIF89aGÕÿ11B99ZRJkcR-œk½Î ½„Æµ{µ„{ZÖÆŒï ÞœçÖ”-{sRskJ„{RkcBœ”k¥{sJµ (CORTADO)

Luego el browser es responsable de interpretar y mostrar en la pantalla el código html y la imagen que recibió del servidor.

Tecnologías disponibles para el desarrollo de aplicaciones:

Para desarrollar aplicaciones y dotar a las páginas web de funcionalidad se puede trabajar tanto en el lado del cliente como en el lado del servidor, las variantes son:

Programación en el cliente:

- El browser envía un request.
- El server envía un response que contiene código que el browser entiende.
- El browser interpreta el código enviado por el server y realiza una determinada acción.

Programación en el servidor:

- El browser envía un request.
- El server ejecuta una aplicación que realiza una determinada acción.
- El server envía el resultado de dicha aplicación al cliente.
- El browser muestra el resultado recibido del server.

Esquema mixto: (programación en el cliente y en el servidor)

- El browser envía un request.
- El server ejecuta una aplicación que realiza una determinada acción.
- El server envía el resultado de dicha aplicación al cliente conteniendo código a interpretar por el browser.
- El browser interpreta el código enviado por el server y realiza una determinada acción.

La programación del lado del cliente tiene como principal ventaja que la ejecución de la aplicación se delega al cliente, con lo cual se evita recargar al servidor de trabajo. El servidor sólo envía el código, y es tarea del browser interpretarlo. La gran desventaja de esta metodología es que el código que el server envía es "sensible" a que cosas puede o no hacer el browser. El usuario puede, por ejemplo, decidir deshabilitar una funcionalidad del browser que es necesaria para que se ejecute un determinado servicio o peor aún, browsers distintos pueden interpretar el mismo código de distintas formas. Típicamente Netscape y Microsoft, que producen los dos browser más usados del mercado, no se ponen de acuerdo sobre como se implementan diversas tecnologías en el cliente.

Programar del lado del servidor tiene como gran ventaja que cualquier cosa puede hacerse sin tener en cuenta el tipo de cliente, ya que la aplicación se ejecuta en el servidor que es un ambiente controlado. Una vez ejecutada la aplicación, el resultado que se envía al cliente puede estar en un formato "normalizado" que cualquier cliente puede mostrar. La desventaja reside en que el server se sobrecarga de trabajo ya que además de servir páginas es responsable de ejecutar aplicaciones. A menudo esto redunda en requisitos de hardware mayores a medida que el server ejecuta más y más servicios.

Programación en el cliente	**Programación en el servidor**
HTML	CGI (Cualquier Lenguaje)
CSS	ASP
DHTML	PHP
JavaScript	mod_perl
Java	
VBScript	

Debido a las incompatibilidades existentes y a la posibilidad de que el usuario controle que cosas se ejecutan y cuales no la programación del lado del cliente no es muy recomendable y debe limitarse a código altamente standard que pueda interpretarse de cualquier forma en cualquier browser, lo cual obliga a ejecutar la gran mayoría de las aplicaciones y servicios de un web site del lado del servidor.

Server Side Programming.

Para el desarrollo de aplicaciones del lado del servidor existen 3 grandes metodologías, utilizar el protocolo CGI, utilizar una API provista por el web-server o bien utilizar un "módulo" del web server.

El protocolo CGI:

El protocolo CGI (Common Gateway Interface) fue creado para establecer un protocolo standard de comunicación entre el web-server y cualquier lenguaje de programación de forma tal que desde el lenguaje "x" puedan recibirse datos que el usuario envía usando el método "POST" o "GET" y además el resultado de la aplicación sea derivado por el web-server al browser. Típicamente para recibir datos se usa alguna biblioteca o módulo del lenguaje elegido que implementa el protocolo CGI y para enviar datos simplemente se envían al standard-output desde el lenguaje elegido y el web-server se encarga de redireccionar esto al browser.

Para ejecutar una aplicación CGI el web-server en general procede de la siguiente manera:

- Se toma el "request del browser" y los datos que se envían al server por método "GET" o "POST" se pasan a variables de ambiente.
- El server redirecciona su salida standard al browser.
- El server crea un proceso (Fork) (que tiene la salida standard redireccionada)
- El server ejecuta en el proceso creado la aplicación deseada.
- Se ejecuta la aplicación

Cuando la aplicación termina de ejecutarse el proceso muere. Dentro de la aplicación se usa algún mecanismo para recuperar los datos enviados por el browser desde las variables de ambiente (todos los lenguajes manipulan variables de ambiente). El protocolo CGI justamente consiste en especificar de que forma los datos enviados por el browser se convierten en variables de ambiente, esto en general es transparente al usuario.

De esta forma pueden realizarse aplicaciones para un web-site en casi cualquier lenguaje, los lenguajes interpretados rápidamente ganaron terreno ya que tienen un ciclo de desarrollo en tiempo inferior a los lenguajes compilados y son más fáciles de debuggear dentro del ambiente CGI.

Los lenguajes no interpretados (C, C++) tienen como ventaja que requieren menos recursos del server al generarse el proceso CGI (no hace falta un interprete) y además suelen ser mucho más veloces en su ejecución (no se necesita interpretar nada), sin embargo el desarrollar y debuggear suelen ser tareas muy complejas y no siempre se justifica el esfuerzo si la aplicación es pequeña. En los comienzos de la web la gran mayoría de las aplicaciones se encontraban en la categoría chica / muy chica por lo que la eficiencia no era un factor importante y por eso los lenguajes compilados no se utilizaron demasiado.

La desventaja de las aplicaciones CGI consiste en que el server debe realizar un fork, y ejecutar la aplicación o bien el interprete de la aplicación, y este ciclo que se cumple cada vez que se ejecuta la aplicación CGI insume muchos recursos y en general es costoso en tiempo para el server. Durante muchos años este esquema no muy eficiente dominó ampliamente el mundo de las aplicaciones Web.

Uso de una API del servidor:

Otra técnica factible consiste en utilizar una API (application programming interface) provista por el web-server para desarrollar aplicaciones, es decir que el web-server provee un lenguaje en el cual se pueden desarrollar aplicaciones. Este esquema, como podemos apreciar, es mucho más eficiente que el anterior ya que el web-server es el encargado de ejecutar las aplicaciones en forma directa sin necesidad de crear un proceso. Las desventajas son sin embargo importantes: en primer lugar las aplicaciones creadas en este marco no son portables ya que sólo pueden ejecutarse en un web-server determinado, esto es una gran desventaja frente a las aplicaciones CGI que podían una vez desarrolladas ejecutarse en cualquier servidor. La segunda gran desventaja es que frecuentemente un error de programación de una aplicación podría ocasionar que el web-server deje de funcionar, genere un error, se cuelgue, pierda memoria u otros problemas. Esto ocasiona que este tipo de aplicación no sea confiable.

Uso de un "módulo del web-server"

La tecnología más reciente para la ejecución de aplicaciones consiste en anexar a un web-server "módulos" que permiten al web-server interpretar un determinado lenguaje. De esta forma se logra eficiencia ya que el server no necesita crear un nuevo proceso por cada aplicación que ejecuta. Las aplicaciones son portables ya que son desarrolladas en un lenguaje standard que no depende del web-server, las aplicaciones son confiables ya que si bien pueden producir un error en el lenguaje en que están diseñadas si el módulo es sólido dichos errores no pueden comprometer al web-server. Al combinar las ventajas más importantes del ambiente CGI y un desarrollo basado en APIs evitando los inconvenientes de los mismos este esquema suele ser el más adecuado para ejecución de aplicaciones.

Cuadro Resumen:

	CGI (interpretado)	**CGI (compilado)**	**API del server**	**Modulo del server**
Ejemplos	Perl, Python	C, C++	Netscape Enterprise	PHP, ASP, Mod_perl, mod_python, FastCGI
Tiempo de desarrollo	Corto	Largo	Medio	Corto
Debugging	Sencillo	Complejo	Complejo	Sencillo
Confiabilidad	Alta	Alta	Baja	Alta
Eficiencia	Baja	Media	Alta	Alta

Benchmarks:

Test 1: 1000 ejecuciones de un programa de 1 línea.

Tecnología	**Tiempo (segundos)**
CGI : C	20,6
CGI : Perl	23,8
CGI : Python	45,2
PHP	16,2
Mod_python	30,0
Mod_perl	16,6
FastCGI	16,4

Test 2: 100000 ejecuciones de un programa de 1 línea.

Tecnología	**Tiempo (segundos)**
CGI : C	2522
CGI : Perl	2886
CGI : Python	5486
PHP	2420
Mod_python	3519
Mod_perl	2117
FastCGI	2120

Test 3: 10000 ejecuciones de un programa de 7000 líneas.

Tecnología	**Tiempo (segundos)**
CGI : C	258
CGI : Perl	963
CGI : Python	978
PHP	304
Mod_python	347
Mod_perl	476
FastCGI	280

En los tests puede verse como las variantes CGI insumen mucho tiempo de "lanzamiento" de la aplicación. Esto se prueba ejecutando muchas veces aplicaciones muy chicas (solo 1 línea de código). De esta forma podemos ver como en este aspecto las tecnologías que no requieren que el server genere un nuevo proceso (php, fastcgi, mod_perl, mod_python) son mucho más eficientes que aquellas que sí lo necesitan.

En el tercer test donde se ejecuta una aplicación de gran tamaño se puede apreciar que la variante "compilada" en CGI es muy eficiente ya que no requiere tiempo alguno de interpretación y el tiempo necesario para generar el proceso es mínimo en comparación con el tamaño de la aplicación. Lo importante de este tercer test es que tanto php, como mod_perl o fastcgi pese a requerir de un interprete son también veloces en este tercer test.

Evolución de los Web-Sites

Desde la aparición de la web distintos "modelos" o prototipos de web-sites fueron siendo predominantes en distintos periodos de tiempo, de acuerdo a las características "comunes" de los sitios predominantes en cada momento podemos diferenciar 3 generaciones de web-sites distintas:

Web-Sites de primera generación:

En un web-site de primera generación las páginas se desarrollaban, se subían al servidor y el servidor se encargaba de enviar las páginas al browser, es un modelo basado en páginas estáticas, en donde predominaba el uso de texto, links a otros sites o a otras páginas del mismo site y listas con "bullets" para enumerar cosas. Frecuentemente se usaban líneas horizontales "rules" para separar contenidos y las páginas de gran extensión vertical con gran cantidad de texto, listas y links eran comunes. Si bien no eran visualmente atractivos estos sites estaban enfocados a funcionar en forma veloz y entregar al usuario gran cantidad de información interrelacionada.

Web-Sites de segunda generación:

La segunda generación de web-sites implicó una revolución en lo visual, a medida que los web-sites se volvían emprendimientos más comerciales que científicos el hecho de "capturar" usuarios se torno una premisa y por ello se le dio gran importancia al aspecto visual. Las páginas con "layouts" visualmente atractivos fueron más y más populares, el uso abundante de imágenes, imágenes animadas y elementos multimedia se volvió común, aparecieron páginas que controlaban los fonts, el estilo y la posición en la que los mismos se ubicaban. La gran mayoría de los sites usaban tablas HTML para controlar la posición exacta en que los elementos aparecerían en el browser y tags nuevos en html como "font" u otros para controla la forma en la cual los mismos se verían. El concepto fue dominar la presentación de la información. Otra característica importante de esta segunda generación de Web-sites fue la aparición explosiva de más y más aplicaciones a medida. Los servicios que ofrecía un site se volvían factores importantes en la atracción de usuarios, chats, foros de discusión, banners, contadores y muchas aplicaciones más empezaron a aparecer y las falencias del protocolo CGI, a medida que las aplicaciones eran mas grandes y la cantidad de usuarios crecía exponencialmente, comenzaron a hacerse notar.

Web-Sites de tercera generación:

La tercera generación de web-sites siguió basada en lo visual, el gran cambio vino en la forma cómo se generaba la información. Las páginas estáticas que dominaban el 100% de los sitios de primera y segunda generación fueron reemplazadas por páginas dinámicas que el web-server generaba en el momento, a partir de información que en general se guardan en una base de datos. Estos sitios "dinámicos" permiten actualizar la información e incluso cambiar completamente la forma en que se muestran dichos datos en velocidades asombrosas. Los sitios de tercera generación facilitaron las aplicaciones interactivas, la información en tiempo real y que día a día se ofrecieran nuevos servicios. Las aplicaciones empezaron a desarrollarse también usando otras tecnologías dejando de lado el protocolo CGI. Aplicaciones en ASP, mod_perl o PHP, mucho más poderosas y eficientes que sus pares CGI, son el standard de este tipo de sitios.

Capitulo 1: Generalidades.

Introducción:

PHP es un lenguaje interpretado diseñado para favorecer el desarrollo de web-stes dinámicos y aplicaciones para web-sites. La distribución más popular de PHP es como módulo para el web-server Apache, aunque puede funcionar con las limitaciones que ya conocemos, como un interprete para ejecutar aplicaciones Cgi en aquellos web-servers que no lo soporten como módulo.

PHP se distribuye en formato open-source y es gratuito, una instalación habitual de PHP consiste en compilar el módulo PHP y luego recompilar el Apache para que utilice el módulo recientemente compilado.

Generalidades:

La característica más importante de PHP es que permite combinar código html y código php en una misma página (de extensión php), por ejemplo:

```
<HTML>
<HEAD><TITLE>Hola</TITLE></HEAD>
<BODY>
Hola esta es una prueba. <BR />
<?php
    print("Hola soy una linea generada en php <BR />");
?>
</BODY>
</HTML>
```

Este ejemplo al guardarse en un archivo de extensión .php es automáticamente parseado por el interprete de php cuando el browser envía un pedido. El ciclo es el siguiente:

- El browser envía un pedido de un archivo con extensión php.
- El server analiza que la extensión del request es .php, obtiene el archivo y lo envía al interprete php.
- El interprete php del web-server parsea el archivo en busca de tags <? ?> y procesa todo lo que se encuentre entre dichos tags (puede haber varias apariciones de los tags en un mismo archivo), todo aquello que esta fuera de los tags se envía al browser sin interpretar.
- El resultado combinado de aquello que no debe interpretarse y el resultado del código interpretado se envía al browser.

En nuestro ejemplo el browser recibiría:

```
<HTML>
<HEAD><TITLE>Hola</TITLE></HEAD>
<BODY>
Hola esta es una prueba. <BR />
Hola soy una línea generada en php <BR />
</BODY>
</HTML>
```

Como podemos ver, es muy sencillo combinar código html y php. Para generar html desde php tenemos las siguientes opciones:

- Usar la función "print de php"
- Usar la función "echo de php"
- Cerrar el tag ?> escribir el código html deseado y volver a abrir el tag <?

La tercera opción es la más eficiente en velocidad cuando el código html que debemos generar es fijo, cuando el código html es dinámico podemos usar una mezcla de print y tags que abren y cierran que suele ser lo mas eficiente, por ejemplo:

```
<form name=”<? print(“$nombre_form”)?>”>
etc…
```

Conceptos básicos en la generación de sites dinámicos con PHP.

Una característica interesante de php es que permite realizar “includes” dentro de un script php, de esta forma se puede modularizar una página o el layout de una página en varios módulos php que se desarrollan en forma independiente, además pueden desarrollarse en php componentes reusables que luego se utilizan en otras páginas o incluso en otros sites.

Una forma común de trabajo usando php para generar sitios dinámicos es definir “templates” o “layouts” en los cuales se divide la página en “zonas” o “módulos” que serán desarrollados en php, el layout de la página con la forma y tamaño de cada zona se puede definir sin problemas usando por ejemplo tablas de html.

A lo largo de este curso desarrollaremos a modo de ejemplo un mini-portal de noticias dinámico al cual le agregaremos servicios o aplicaciones a medida que se estudian distintas características de php. Supongamos que tenemos por el momento un único “template” o “layout” para nuestro sitio que determina la forma en la cual se vera la “home page” del mismo, el equipo de diseño nos entrega el siguiente layout:

<table>
<tr><td>LOGO</td><td colspan="4">BANNER</td></tr>
<tr><td colspan="5">BUSCADOR</td></tr>
<tr><td>BOTON S1</td><td>BOTON S2</td><td>BOTON S3</td><td colspan="2">BOTON S4</td></tr>
<tr><td colspan="2">Barra de links y Aplicaciones</td><td colspan="3">CONTENIDOS</td></tr>
<tr><td colspan="5">Información de copyrigh, y pie de página</td></tr>
</table>

Como podemos ver el site cuenta con 4 secciones que se acceden desde una barra navegadora ubicada debajo del search box, además existe una barra de links y servicios a la izquierda y una zona de contenidos que es la zona principal de la página.

Aun sin saber que funcionalidad tiene o de que forma se debe implementar cada parte podemos esquematizar el layout de la página usando php y html de la siguiente forma:

```
<HTML>
<HEAD>
<TITLE>Layout</TITLE>
```

```
</HEAD>
<BODY>
<table width="640" border="1">
<tr><td width="20%">Logo</td><td colspan="3">Banner</td></tr>
<tr><td colspan="4">buscador</td></tr>
<tr><td>s1</td><td>s2</td><td>s3</td><td>s4</td></tr>
<tr><td width="20%">ColIzq</td><td colspan="3">Contenidos</td></tr>
</table>
</BODY>
</HTML>
```

Luego podemos reemplazar cada "zona" de la home page por un include en php que generara dinámicamente la parte de la página en cuestión:

```
<HTML>
<HEAD>
<TITLE>Layout</TITLE>
</HEAD>
<BODY>
<table width="640" border="1">
<tr><td width="20%"><? Include("logo.php");?></td><td
colspan="3"><?include("banner.php");?></td></tr>
<tr><td colspan="4"><?include("buscador.php");?></td></tr>
<tr><?include("botonera.php");?></tr>
<tr><td width="20%"><?include("izq.php");?></td><td
colspan="3"><?include("contenidos_home.php");?></td></tr>
</table>
</BODY>
</HTML>
```

De esta forma hemos modularizado el layout de la página y tenemos como resultado que deben desarrollarse los siguientes módulos:

- logo.php
- banner.php
- buscador.php
- botonera.php
- izq.php
- contenidos_home.php

Funcionalidad de PHP:

- Funciones de calendario y manipulación de calendarios usando MCAL
- Programación orientada a objetos
- Funciones para creación de archivos PDF
- Funciones de manejo de cajeros cybercash
- Parser de documentos XML
- WDDX
- Funciones de compresión de datos
- Manejo de archivos DBM
- Funciones para manipulación de fechas
- Funciones para manejo de directorios
- Funciones de encripción de datos
- Funciones de acceso al filesystem
- Funciones para manejo de FTP
- Funciones de hashing
- Generación dinámica de imágenes
- Manejo de cuentas de mail IMAP y POP3
- Funciones para envió de mail
- Funciones de networking usando sockets
- Funciones matemáticas
- Serialización de estructuras de datos
- Acceso a bases de datos (Mysql, Oracle, Postgress, Sybase, etc)
- Manejo de expresiones regulares.
- Manejo de sesiones.

Capítulo 2: Introducción al lenguaje.

PHP es un lenguaje no posicional, por lo que no importa la columna en la cual se comience a escribir el código. Tampoco influye sobre el código la cantidad de saltos de línea (enter) que se coloquen, ni la cantidad de espacios.

La forma en la que se separan las distintas sentencias es mediante la utilización de ";". En PHP cada sentencia debe finalizar con ";".

Se puede escribir más de una sentencia en la misma línea siempre y cuando las mismas se encuentren separadas con ";".

Comentarios:

En PHP hay 3 formas distintas de incluir comentarios:

```
/* Al estilo de C
   en donde el comentario empieza
   y termina delimitado por barra asterisco y asterisco barra
*/

O bien usando
// Comentario

O por último
# Comentario
```

En las dos últimas variantes el comentario empieza en donde se encuentra el "//" o el "#" y termina cuando termina la línea.

Tipos de Datos:

PHP soporta los siguientes tipos de datos:

- Enteros
- Vectores
- Binarios de punto flotante
- Strings
- Objetos

En general el tipo de dato de una variable no es decidido por el programador sino que lo decide el lenguaje en tiempo de ejecución, la instrucción settype puede usarse para forzar el tipo de dato de una variable en los raros casos en que esto sea necesario. Todas las variables en php se denotan utilizando el signo '$' precediendo al nombre de la variable.

Enteros:

```
$a = 1234; # número decimal
$a = -123; # número negativo
$a = 0123; # número octal  (83 en decimal)
$a = 0x12; #número en hexadecimal (18 decimal)
```

Flotantes:

Los números de punto flotante pueden notarse de la siguiente manera:

```
$a = 1.234;
$a = 1.2e3;
```

Strings:

En PHP los strings tienen un manejo similar al utilizado en “C” o “C++”, están predefinidos los siguientes caracteres especiales:

\n	Nueva línea
\r	Salto de carro (carring return)
\t	Tabulación
\\	Barra invertida
\$	Signo pesos
\”	Comillas doble

Un string puede inicializarse usando comillas dobles o comillas simples. Cuando se utilizan comillas dobles el interprete de php parsea previamente el string, es decir que reemplaza los nombres de variables que encuentra en el string por el contenido de la variable. Cuando se usan comillas simples el string se imprime tal y como figura sin ser parseado.

Ej:

```
$x=”Juan”;
$s=”Hola $x”;
$t=’Hola $x’
```

$s vale “Hola Juan” y $t vale “Hola $x”.

Otra forma de inicializar un string es usando un string multilinea de la siguiente manera:

```
$str=<<<EOD
Este es un ejemplo de un string
que ocupa varias líneas y se puede
definir así
EOD;
```

Se pueden concatenar strings usando el operador “.” de la siguiente manera:

```
$x=”hola”;
$y=”mundo”;
$s=$x.$y; #$s es igual a “holamundo”.
$s=$x.” “.$y; # Aquí $s vale “hola mundo”
```

Vectores:

Los vectores en php actúan tanto como vectores tradicionales (indexados por número) así también como vectores asociativos (indexados por clave).

Los vectores pueden crearse usando las instrucciones “list” o “array” o bien inicializando en forma explícita cada elemento del vector.

```
$a[0]=”hola”
$a[1]=”mundo;
$a[“clave”]=”valor”;
```

Utilizando la notación especial $v[]; se pueden agregar elementos a un vector.

```
$a[0]=”nada”;
$a[1]=”hola”;
$a[]=”mundo”;  #Asigna a $a[2] el valor “mundo”.
```

Existen funciones especiales para ordenar vectores, contar la cantidad de elementos de los mismos, recorrerlos, etc. (Ver el capítulo sobre vectores)

Matrices:

La definición, inicialización y uso de matrices en PHP es sencilla. Se puede pensar una matriz en PHP como un vector de vectores, por lo que se puede utilizar la misma lógica que en los primeros.

```
$a[0][1]=”Hola”;
$a[0][“clave”]=”una cosa”;
$a[“clave1”][“clave2”][0][1]=”otra cosa”;
etc…
```

Para incluir el valor de un elemento de un vector en un string se deben usar llaves para indicar el alcance del nombre de la variable a reemplazar:

```
Echo “Esta es una prueba {$a[0][1]}”;
```

Una forma útil de inicializar un vector asociativo es usando la siguiente notación:

```
$a=array(
  “color” => “rojo”,
  “edad” => 23,
  “nombre” => “juan”
);
```

Para crear una matriz se pueden anidar las declaraciones de tipo array.

```
$a = array(
       "apple"  => array(
            "color"  => "red",
            "taste"  => "sweet",
            "shape"  => "round"
       ),
       "orange"  => array(
           "color"  => "orange",
           "taste"  => "tart",
            "shape"  => "round"
       ),
       "banana"  => array(
            "color"  => "yellow",
            "taste"  => "paste-y",
            "shape"  => "banana-shaped"
       )
  );
```

Constantes:

Para definir una constante se utiliza la instrucción “define” de la forma:

define(“PI”,3.14151692);

Luego las constantes pueden usarse como variables tradicionales ($PI) con la salvedad de que no se les puede asignar un valor.

Operadores:

Los operadores aritméticos en PHP también se asemejan al C:

```
$a + $b;        //suma
$a - $b;        //resta
$a++;           //pos-incremento, esta sentencia devuelve el valor de $a y lo incrementa en 1.
++$a;           //pre-incremento, incrementa en 1 el valor de $a y devuelve el valor incrementado.
$a--;           //pos-decremento
--$a;           //pre-decremento
$a * $b;        //multiplicación
$a / $b;        //división
$a % $b;        //módulo
```

Asignación:

La asignación se resuelve con el signo igual (“=”).

```
$a=5;           //Asigna 5 a $a
$a=$b;          //Asigna el valor de $b a $a
$b=($c=6);      //Asigna 6 a $c y a $b
```

Y pueden combinarse asignación y operadores de la forma:

```
$a+=5;          //Suma y asigna 5 a $a
$x.=”hola”;     //Concatena $x con “hola”
```

Operaciones con bits:

```
$a & $b;        //$a AND $b
$a | $b;        //$a OR $b
$a ^ $b;        //$a XOR $b
~$a;            //NOT $a
$a << $b;       //Shift hacia la izquierda $b posiciones
$a >> $b;       //Shift hacia la derecha $b posiciones
```

Comparaciones:

```
$a = = $b;      //true si $a igual a $b
$a = = = $b;    //true si $a igual a $b y además son del mismo tipo
$a >= $b;       //mayor o igual
$a <= $b;       //menor o igual
$a != $b;       //true si a y b son distintos
```

Operador @

Cuando se antepone @ a una expresión se suprimen los errores que la expresión pudiera generar.

Ej:

@($c=$a/$b);
Operador de ejecución:

PHP soporta el uso de “backticks” (comillas invertidas) para ejecutar un comando desde el shell y devolver el resultado de la ejecución del comando en una variable:

$result=`ls –l`;

Operadores lógicos:

```
$a && $b;       //True si $a es true y $b es true
$a || $b;       //True si $a es true o $b es true
$a xor $b;      //Or exclusivo
!$a;            //True si $a es falso (NOT)
```

Estructuras de Control:

If:

```
if (expresión) sentencia;

if (expresión) {sentencias;}

if (expresión) {
  sentencias;
} else {
  sentencias;
}

if (expresión) {
  sentencias;
} elseif (expresión) {
  sentencias;
} else (expresión) {
  sentencias;
}
```

While:

```
while (expresión) {
 sentencias;
}

do {
 sentencias;
} while(expresión)
```

For:

```
for (expr1,expr2,expr3) {
 sentencias;
}
```

La primera expresión cumple la función de inicializar las variables de control del FOR. Esta expresión se cumple incondicionalmente, más allá de que se entre dentro del ciclo o no.
La expresión 2 se evalúa siempre que se este por ingresar al ciclo del FOR, aún cuando se ingresa al for por primera vez.
La tercera expresión se ejecuta cada vez que se termina el ciclo. Por lo general se utiliza esta expresión para indicar el incremento de alguna variable que se este utilizando para el FOR. La ejecución de esta expresión es también incondicional y es la que se ejecuta inmediatamente antes de evaluarse la expresión 2.

Ejemplo:

```
for ($i=0;$i<5;$i++) {
 print("$i");
}
```

Foreach:
Para realizar ciclos con la cantidad de ocurrencias en un vector se utiliza el comando foreach:

```
foreach ($vector as $variable) {
 sentencias;
}

foreach ($vector as $clave => $valor) {
 sentencias;
}
```

Por cada iteración cada elemento de $vector es asignado a $variable.
El segundo caso es aplicable para recorrer vectores asociativos.

Break:

Break permite salir del ciclo actual “for” , “while” o “switch”

Ejemplo:

```
for ($i=0;$i<6;$i++) {
 If($i==$b) break;   //Sale del ciclo si $i es igual a $b
}
```

Switch:

El switch permite ejecutar un grupo de sentencias de acuerdo al valor de una variable:

```
switch ($variable) {
 case valor:
    sentencias;
    break;
  case valor2:
    sentencias;
    break;
  default:
    sentencias;
    break;
}
```

Cuando el valor de la variable ($variable) coincide con el valor de algún “case”, se ejecutan las sentencias que se encuentran a continuación.

En este caso se utiliza la sentencia “break” en forma prácticamente obligatoria, porque en caso de no existir esta sentencia se seguiría ejecutando linealmente todas las sentencias continuas, es decir las sentencias de los demás “cases” inferiores.

Por último, la opción “default” se utiliza generalmente para cuando el valor de la variable no coincide con ningún “case”. Estas sentencias se ejecutan siempre, salvo en el caso de que se ejecute antes un “break”.

Funciones:

Definir subrutinas (funciones) en php es sencillo:

```
function prueba($a,$b) {
  $r=$a + $b;
  return $r;
}
```

Para invocar a la función basta con hacer $x=prueba(4,6);
Los parámetros que recibe la función pueden ser enteros, flotantes, strings, vectores u objetos es decir cualquiera de los tipos de datos soportado por PHP.
El valor devuelto por la función también puede ser cualquier tipo de datos de php.

Parámetros default:

Es posible asignar un valor default a los parámetros que recibe una función de forma tal que cuando se invoca la función si se ignora el parámetro el mismo es asignado al default.

```
function prueba($a=2,$b=3,$c=5) {
  //código
}
```

Si se llama prueba(4) // Entonces $a=4, $b=3 y $c=5

Se puede saber cuantos parámetros recibió una función usando func_num_args() y se puede obtener el iesimo parámetro de una función con func_get_arg(número_de_parámetro);

Finalmente los nombres de funciones pueden guardarse en variables e invocarse las mismas usando el nombre guardado en una variable.

```
Ej: $nombre=sumar;
$nombre(4,5);      //Llama a la función sumar
```

Esto permite guardar nombres de funciones en tablas, archivos, vectores etc lo cual da lugar a ciertas maniobras interesantes que no parece demasiado conveniente enumerar en este capítulo.

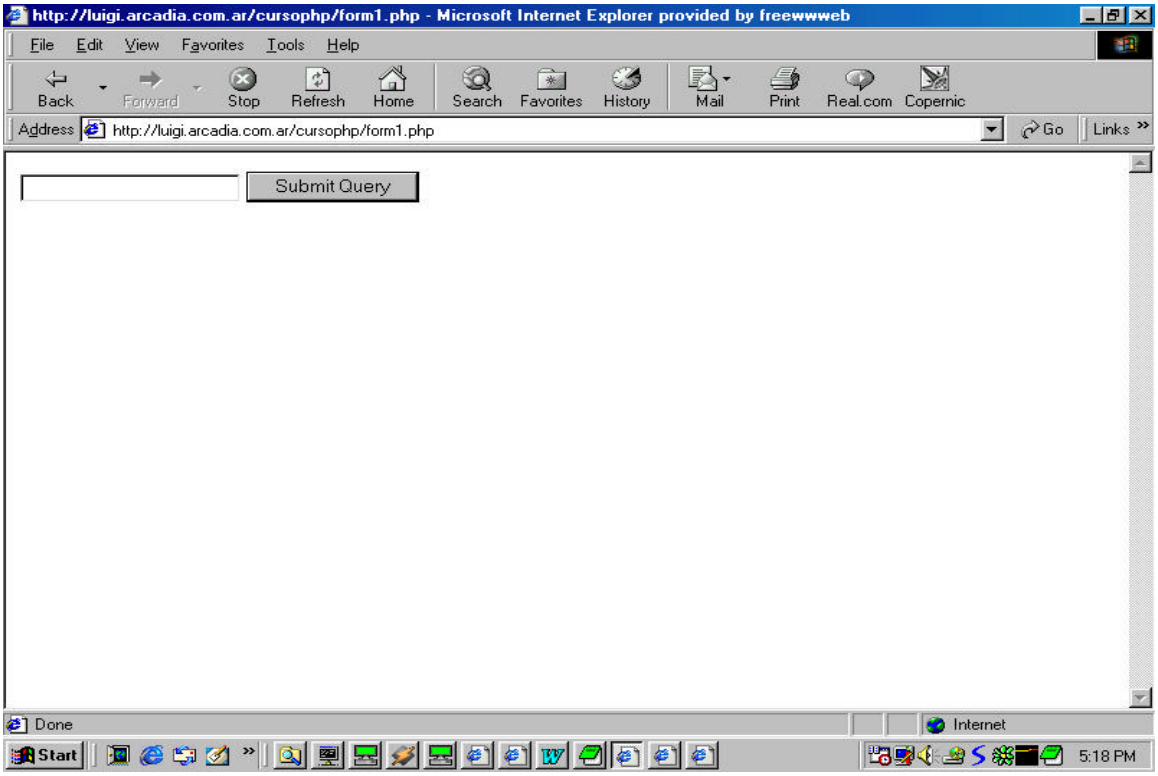
Capítulo 3: Manejo de Forms.

El mecanismo básico de interacción entre el usuario y un web-site esta dado por el uso de formularios html, el server envía un formulario que el browser muestra en pantalla permitiendo al usuario ingresar datos, luego los datos en el formulario viajan al server en el próximo request realizado por el browser para ser procesados en el mismo. La respuesta del server suele depender de los datos recibidos en el formulario.

El siguiente es un ejemplo de formulario en HTML usando un campo de entrada de tipo “text”

```
<FORM ACTION=”procesar.php”  METHOD=”POST”>
<INPUT TYPE=”text” NAME=”texto”>
<INPUT TYPE=”submit” NAME=”proc”>
</FORM>
```

Este formulario HTML se ve en pantalla de la siguiente manera:



Una vez que el usuario ingresa un texto y presiona el botón de submit el browser genera un request con método “Post” al script “procesar.php” que es el script que se va a encargar de procesar los datos ingresados en el formulario.

Dentro del script php los datos del formulario se reciben en variables php que tienen el mismo nombre que los indicados con “NAME” en el formulario, en este caso el script recibe $texto con el texto tipeado por el usuario en el formulario.

El script que recibe el formulario podría por ejemplo ser:
(procesar.php)

```
<?
print (“El valor ingresado en el formulario es: $texto <BR />”);
?>
```

En PHP es posible que un form se procese a si mismo, esto lo podemos hacer de la siguiente manera:
(form1.php)

```
<?
if(isset($proc)) {
  print(“el valor ingresado es: $texto”);
} else {
?>

<FORM ACTION=”form1.php”  METHOD=”POST”>
<INPUT TYPE=”text” NAME=”texto”>
<INPUT TYPE=”submit” NAME=”proc”>
</FORM>

<?
} //Esto cierra el else que abrimos arriba.
?>
```

Notar que el nombre del script que muestra el formulario es el mismo que el script usado en “action” para procesarlo, la instrucción isset de PHP devuelve true si la variable esta seteada, para un formulario si el usuario presiona el botón de submit se setea automáticamente la variable que corresponde al “NAME” del botón submit del formulario, por eso preguntamos si esta seteado $proc para saber si hay que mostrar el formulario o procesarlo. Podría también procesarse el formulario y a su vez mostrarlo o mostrar otro distinto, las variantes dependen de que es lo que se quiere hacer.

Text type:

Para ingresar texto mediante un formulario html se usa el tag input con atrobuto type=”text”, los atributos disponibles son:

Atributos:

maxlength=”numero”	Cantidad máxima de caracteres que se pueden ingresar
name=”text”	Nombre de la variable php que recibirá el valor
size=”numero”	Tamaño del campo de entrada a mostrar en pantalla
value=”texto”	Valor inicial a mostrar en el campo de entrada (default)

Hidden Type:

El tag input con type=”hidden” funciona en forma idéntica a un tipo “text” con la salvedad de que no se muestra en pantalla, esto es útil para pasar variables entre formularios o guardar variables “ocultas” en un formulario.

Checkboxes:

Los checkboxes son campos de entrada que soportan solamente los estados de seteado o no. Para ello se usa el tag input con type=”checkbox” , los atributos disponibles son:

Atributos:

Checked	Si el atributo esta presente el checkbox aparecerá marcado por default.
name=”text”	Nombre de la variable php que recibe el valor.
value=”text”	Valor que toma la variable si esta seteada, el default es “on”

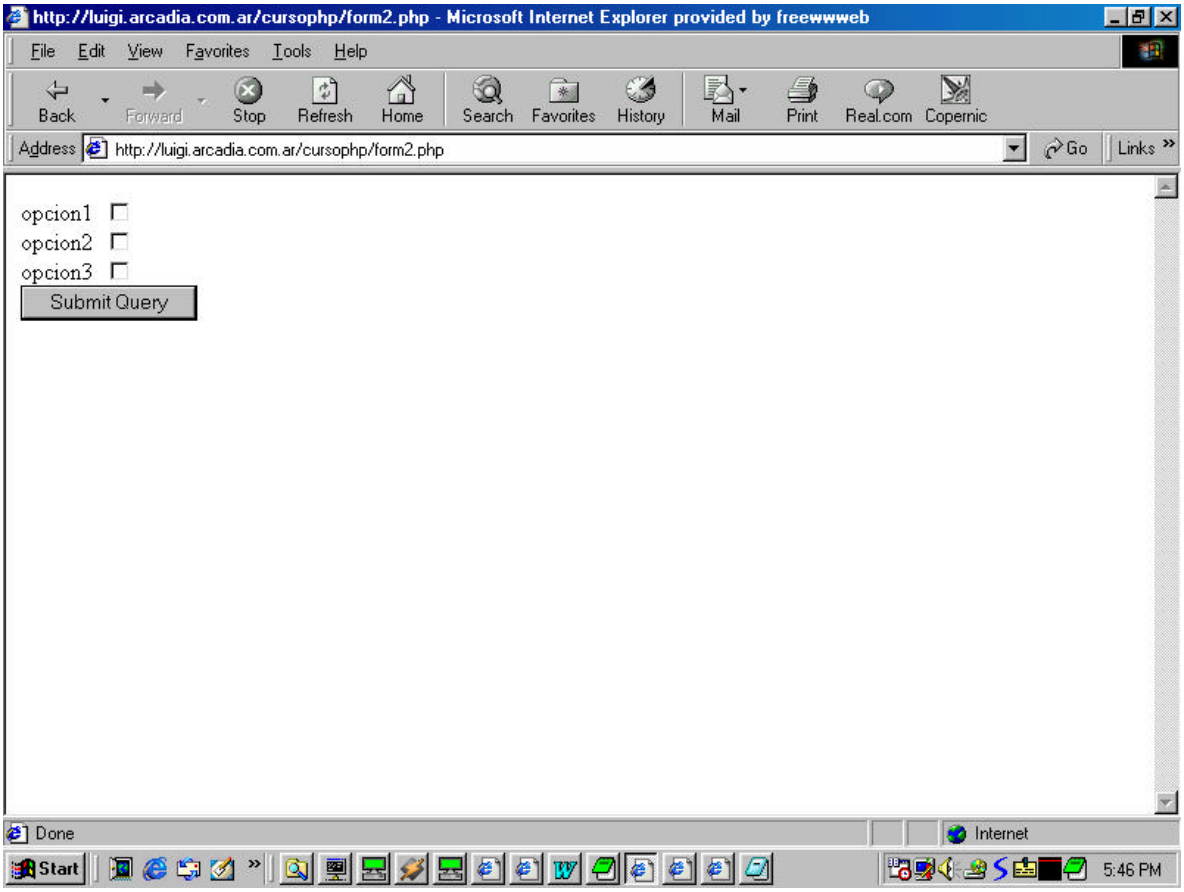
El script que recibe los resultados sólo recibe los nombres de los checkboxes que están seteados, es común en php generar una lista de checkboxes a partir de un vector, veamos un ejemplo:

(form2.php)

```
<FORM ACTION=”form2.php” METHOD=”post”>
<?
$vector=array(“opcion1”,”opcion2”,”opcion3”);
for ($i=0;$i<count($vector);$i++) {
  print(“$vector[$i]”);
  ?>
 <input type=”hidden” name=”valor[<?print($i);?>]”
value=”<?print(“$vector[$i]”);?>”>
  <input type=”checkbox” name=”vector[<?print($i);?>]”> <br>
  <?
}
?>
<INPUT TYPE=”submit” name=”proc”>
</FORM>
```

Este es un ejemplo muy útil en el cual el formulario html no es siempre el mismo sino que es generado dinámicamente desde php en base a por ejemplo el contenido de un vector.

El formulario se muestra en el browser de la siguiente manera:



Y el código html que recibe el browser para mostrar el formulario (que se genera en el servidor) es:

```
<FORM ACTION="form2.php" METHOD="post">
opcion1 <input type="hidden" name="valor[0]" value="opcion1">
<input type="checkbox" name="vector[0]"> <br>
opcion2 <input type="hidden" name="valor[1]" value="opcion2">
<input type="checkbox" name="vector[1]"> <br>
opcion3 <input type="hidden" name="valor[2]" value="opcion3">
<input type="checkbox" name="vector[2]"> <br>
<INPUT TYPE="submit" name="proc">
</FORM>
```

Observar el uso de campos de texto ocultos para indicar cual es el valor de un textbox en caso de estar seleccionado, también podríamos haber usado el campo value de los checkboxes, es otra forma de hacer lo mismo. Le podemos agregar al formulario la opción de mostrar cuales son los checkboxes seleccionados usando: (Agregar este código al principio de form2.php)

```
<?
if(isset($proc)) {
  for ($i=0;$i<count($valor);$i++) {
    if(isset($vector[$i])) {
      if($vector[$i]==”on”) {
          print(“$valor[$i] viene seleccionado”);
     }
    }
  }
}
?>
```

Como resultado el script informa cuales son los checkboxes que han sido seleccionados por el usuario y cuales no. El script completo es:

(form2.php)

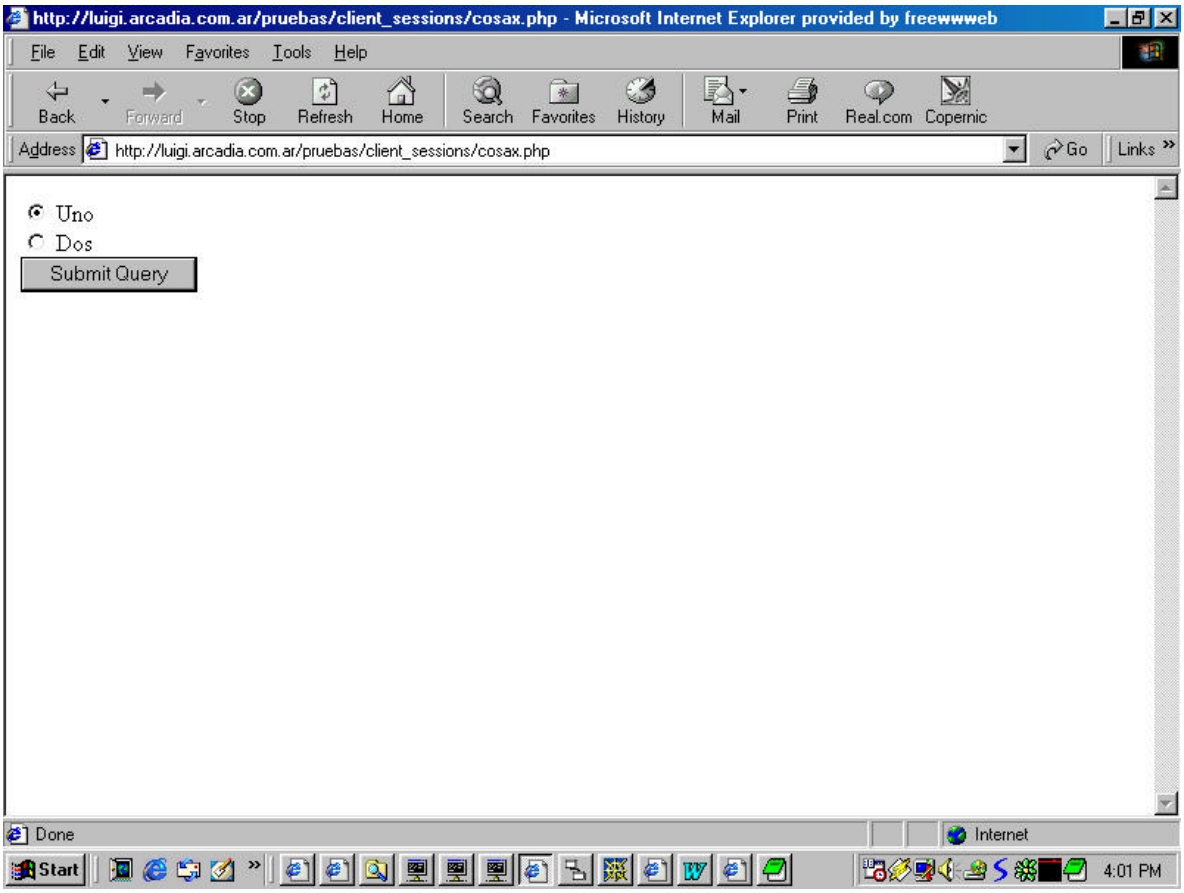
```
<?
if(isset($proc)) {
  for ($i=0;$i<count($valor);$i++) {
    if(isset($vector[$i])) {
      if($vector[$i]=="on") {
          print("$valor[$i] viene seleccionado");
     }
    }
  }
}
?>

<FORM ACTION="form2.php" METHOD="post">
<?
$vector=array("opcion1","opcion2","opcion3");
for ($i=0;$i<count($vector);$i++) {
  print("$vector[$i]");
  ?>
 <input            type="hidden"            name="valor[<?print($i);?>]"
value="<?print("$vector[$i]")
;?>">
  <input type="checkbox" name="vector[<?print($i);?>]"> <br>
  <?
}
```

Radio Buttons:

Un radio button también es un tipo de botón de 2 estados (encendido-apagado) pero estos pueden agruparse de forma tal que sólo un radio-button del grupo pueda estar prendido. El nombre de radio button fue puesto por su funcionamiento parecido a los botones de las viejas radios de automóviles, que siempre debía estar presionado uno de los botones y nunca (mientras funcionaba correctamente) podían estar dos presionados al mismo tiempo.

Los atributos son checked, name y value. Todos los radio buttons del mismo nombre pertenecen al mismo grupo, es decir que de todo el grupo sólo uno podrá ser seleccionado. El value del botón determina cual es el radio elegido del grupo. Ejemplo:

```
<?
 if(isset($proc)) {
   print("El radio seteado es $grupo <BR />");
 }
?>

<FORM ACTION="form3.php" METHOD="post">
<input type="radio" name="grupo" value="uno" checked> Uno <br>
<input type="radio" name="grupo" value="dos"> Dos <br>
<INPUT TYPE="submit" name="proc">
</FORM>
```



El formulario que se despliega en pantalla es:

Y solamente uno de los dos radio buttons puede estar habilitado.

Image:

Es posible reemplazar el boton submit por una imagen gif o jpg de forma de darle al botón el look and feel que se quiera, esto se hace con el tag input type=”image” los atributos son:

name=”texto”	Cumple la misma función que el atributo name de submit.
src=”url”	URL de la imagen a mostrar, ejemplo: “images/boton.jpg”

TextArea:

Un textarea permite ingresar texto en una caja de formato mayor a un input type=”text” , la notación de un textarea es distinta a la de un tag de input. Los atributos son los siguientes:

cols=”numero”	Número de columnas del textarea
rows=”numero”	Número de líneas
name=”texto”	Nombre de la variable que recibe el texto
wrap=”off/virtual/physical”	Forma en la cual se cortan las palabras cuando termina cada línea, off elimina el corte de palabras, virtual muestra los cortes pero estos no son transmitidos al server, physical corta las palabras y además transmite los saltos de línea al server.

Ejemplo:

<textarea name=”texto” cols=”20” rows=”10”>
</textarea>

Entre el tag que abre y cierra si se desea se puede poner texto que se muestra en el textarea y el usuario puede modificar.

File Uploads

El último tipo de dato que se puede transferir al server usando un formulario es un archivo, este es un tipo de transferencia especial pues implica generar un archivo en el file-system del web-server a partir de un archivo que el usuario selecciona desde su disco local.

HTML soporta el upload de archivos usando el tag <input> con type=”file”, este tipo de input genera un boton “browse” en el browser que permite al usuario seleccionar un archivo desde su file-system local (usando una caja de navegación por los discos standard del sistema operativo).

El formulario para subir un archivo es de la forma:

```
<FORM ENCTYPE="multipart/form-data" ACTION="upload.php" METHOD=POST>
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <INPUT NAME="userfile" TYPE="file">
<INPUT TYPE="submit" VALUE="Send File">
</FORM>
```

Como puede verse hay un campo oculto que indica cual es el limite máximo de tamaño que se puede subir, este valor es chequeado en el “cliente”, además PHP dispone de una variable que se inicializa en el archivo de configuración de php (en gral /var/lib/php.ini) allí se limita el tamaño máximo de los uploads al llegar al “server”

El script upload.php que recibe los datos del formulario recibe las siguientes variables:

- $userfile – Path del archivo almacenado en el server.
- $userfile_name – Nombre del archivo segun el usuario
- $userfile_size – Tamaño del archivo subido
- $userfile_type – Mime type del archivo, por ejemplo image/gif

El script que recibe el archivo es responsable de hacer lo que corresponda con el mismo ya que en general el archivo se almacena en un directorio temporal y es eliminado una vez que termina el script. El script debe almacenar el archivo en una base de datos, moverlo a un directorio permanente, tomar datos de el o realizar el procesamiento que corresponda.

Capitulo 5: Manejo de Bases de Datos (MySQL)

Una de las características más importantes de PHP es su integración con diversos motores de base de datos,. El PHP está construido para generar en forma sencilla páginas web dinámicas a partir de información almacenada en bases de datos. A continuación mostramos las funciones más importantes y ejemplos típicos de uso para una base MySql.

1. Conexión a la base

$db_link=mysql_connect(hostname, user, password);

Ejemplo:

$db=mysql_connect("localhost","root","secreto");

La función realiza la conexión a la base Mysql y devuelve "false" si hubo algún error en la conexión o un link a la conexión a la base en caso de que la conexión sea exitosa.

El link es un número que indica la sesión dentro del MySql. Esta sesión se mantiene hasta que [...]. Para finalizar la conexión se debe utilizar la función mysql_close(). Es muy importante cerrar la conexión a la base de datos una vez finalizadas las transacciones para evitar la sobrecarga en el motor de la base de datos.

2. Selección de la base de datos a utilizar

mysql_select_db(database_name, db_link);

Ejemplo:

mysql_select_db("test",$db);

Esta función configura cual es la base de datos que se utilizara por omisión. En este caso el link a utilizar en esta función es el link que se obtuvo al ejecutar la función mysql_connect.

La función mysql_select_db devuelve el valor "false" en caso de que se encuentre algún error, como por ejemplo la inexistencia de la base de datos.

En este punto cabe aclarar que la denominación de las bases de datos de MySql es case-sensitive, por lo que debemos mantener un standard a la hora de elegir los nombres de las distintas bases de datos.

3. Queries a la base de datos.

$result=mysql_query(query,db_link);

Ejemplo:

$result=mysql_query("update clientes set deudor='si' where apellido='Picapiedras', $db)

$query="insert into clientes (nombre, Apellido) values (Pedro, Mármol) ";
$result=mysql_query($query,$db);

Nuevamente el link que se debe usar es el que se obtiene al conectarse a la base, mysql_query devuelve falso en caso de que el query no pueda ejecutarse (error de SQL) o bien un result set en los casos que devuelva algún tipo de datos como por ejemplo en un select..

Es muy importante fijarse con que usuario se realizó la conexión a la base de datos a la hora de ejecutar el mysql_connect, ya que la gran mayoría de los errores producidos en esta instancia son el resultado de la falta de permisos para realizar la consulta.

4. Cantidad de Filas Consultadas o Modificadas

4.1 Filas Consultadas

$cantidad=mysql_num_rows($result);

Ejemplo:
$query=”select nombre, telefono from contactos where edad between 20 and 25 and sexo=’f’”;
$result=mysql_query($query,$db);
$cant=mysql_num_rows($result);

Esta función devuelve la cantidad de filas que se obtuvieron luego de ejecutar una instrucción de consulta como por ejemplo la función select.
En el caso del ejemplo, en la variable $cant nos dirá cuantas chicas de entre 20 y 25 años tenemos en nuestra lista de contactos.

4.2 Filas Modificadas

$cantidad=mysql_affected_rows(db_link);

Ejemplo:

$cuantos=mysql_affected_rows($db);

Devuelve cuantos registros fueron afectados por un query con insert, update, o delete, notar que se le pasa el db_link ya que el result_set no tiene sentido. Si el query fue un delete sin clausula “where” esta función devuelve cero independientemente del número de registros eliminados de la tabla.

5. Obtención de registros de una consulta

5.1 Obtención de datos en un result set

$var=mysql_fetch_row(result_set);

Ejemplo:
$query=”select nombre, telefono from contactos where edad between 20 and 25 and sexo=’f’”;
$result=mysql_query($query,$db);$rs=mysql_fetch_row($result);

Toma un registro del result set y lo devuelve en un vector en el cual el elemento con índice 0 es la primer columna del registro, el elemento con índice 1 es la segunda columna, etc. Si no hay más registros por devolver devuelve false.
Los valores de los campos solicitados en el result set son devueltos en forma de array, es decir que para acceder al valor de nombre de la primera fila de nuestro ejemplo debo llamar a la variable $rs[0].Ejemplo:

```
while($v=mysql_fetch_row($result)) {
  print(“Columna 0: $v[0] Columna 1: $v[1]”);
}
```

5.2 Obtención de datos en un vector

$query=”select nombre, telefono from contactos where edad between 20 and 25 and sexo=’f’”;
$result=mysql_query($query,$db);
$rs=mysql_fetch_array($result);

Ejemplo:

$rs=mysql_fetch_array($result);

Funciona de forma idéntica a mysql_fetch_row pero devuelve los resultados en un vector asociativo indexado por nombre de columna, por lo que para obtener los nombres de nuestra consulta es necesario llamar al vector de la forma $rs[“nombre”]

Ejemplo:

```
while($v=mysql_fetch_array($result)) {
  print(“Columna 0: $v[“nombre”]”);
}
```

6. Cerrar la conexión

mysql_close(db_link)

Por último, esta función nos sirve para cerrar la conexión a la base se usa.

Otras funciones:

Función	Descripción
result=mysql_create_db(database_name,db_link);	Crea una base de datos con el nombre indicado, devuelve true o false según el resultado.
Result=mysql_data_seek(result_set, position);	Mueve el puntero interno de cada result set a la posición indicada (el primer registro es la posición 0), de esta forma la próxima llamada a mysql_fetch_row o mysql_fetch_array devolvera el registro que se acaba de apuntar. Devuelve true o false.
Result=mysql_db_query(database_name, query, db_link);	Realiza una consulta a una base indicada, es obviamente reemplazable por un mysql_select_db y un mysql_query aunque resulta útil si hay que consultar tablas en varias bases distintas.
Result=mysql_drop_db(database_name,db_link);	Dropea la base con el nombre indicado, devuelve true o false.
Object=mysql_fetch_field(result_set, numero_columna);	Devuelve un objeto con información sobre la columna indicada de un result set (0 es la primera columna, 1 la segunda, etc). Las propiedades que se setean en el objeto son: • name – nombre de la columna • table – nombre de la tabla • max_length – longitud maxima de la columna • not_null – 1 si la columna no puede ser null • primary_key - 1 si la columna es primary key • unique_key - 1 si la columna es unique key • multiple_key - 1 si la columna no es unique key • numeric - 1 si la columna es numerica • blob - 1 si la columna es un BLOB • type – Tipo de la columna • unsigned - 1 si la columna es sin signo • zerofill - 1 si la columna es zero-filled
object=mysql_fetch_object(result_set);	Similar a mysql_fetch_array con la diferencia de que los resultados se devuelven en un vector en lugar de un vector asociativo. Luego se puede acceder a las distintas columnas del registro devuelto como propiedades (data_members) del objeto devuelto.
int=mysql_num_fields(result_set);	Devuelve el numero de columnas en un result set.

Manejo de errores:

Mensaje=mysql_error();

Devuelve un texto con el mensaje correspondiente al error que ocurrió en caso de que se produzca un error en la base de datos.

Ejemplo:

```
$result=mysql_query($query,$db);
if(!$result) {
  $x=mysql_error();
  die(“ Ocurrio un error en la base de datos: $x”);
}
```

Generación dinámica de páginas a partir de una base de datos:

Repasemos el layout de nuestro web-site ejemplo:

<table>
<tr><td>LOGO</td><td colspan="4">BANNER</td></tr>
<tr><td colspan="5">BUSCADOR</td></tr>
<tr><td>BOTON S1</td><td>BOTON S2</td><td>BOTON S3</td><td colspan="2">BOTON S4</td></tr>
<tr><td colspan="2">Barra de links y Aplicaciones</td><td colspan="3">CONTENIDOS</td></tr>
<tr><td colspan="5">Informacion de copyrigh, y pie de página</td></tr>
</table>

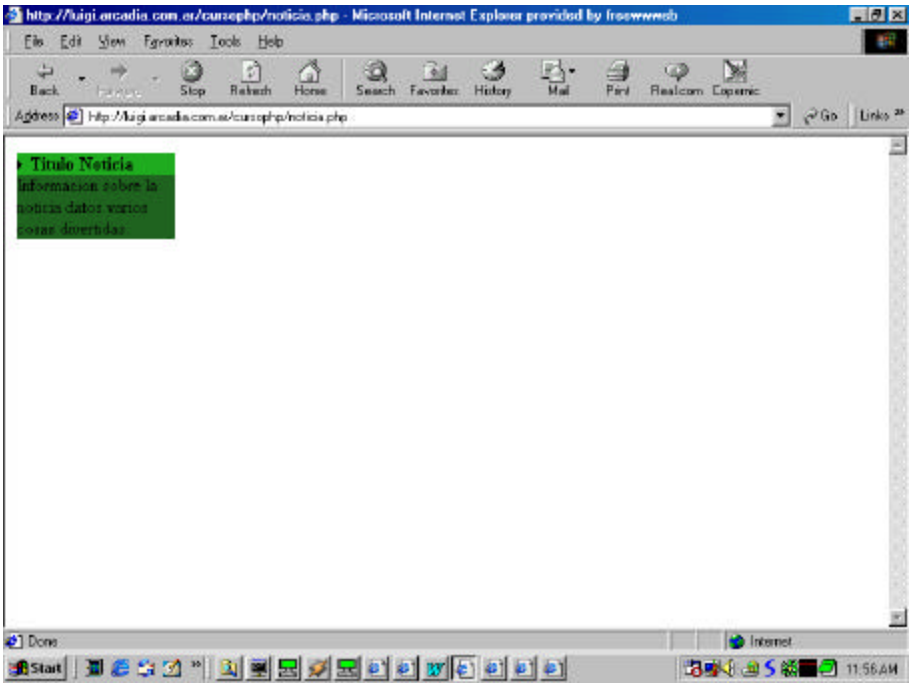
Supongamos que a la barra de links y aplicaciones a la derecha se decide agregarle un módulo “noticia de la semana” el layout queda entonces de la siguiente manera:

<table>
<tr><td>LOGO</td><td colspan="4">BANNER</td></tr>
<tr><td colspan="5">BUSCADOR</td></tr>
<tr><td>BOTON S1</td><td>BOTON S2</td><td>BOTON S3</td><td colspan="2">BOTON S4</td></tr>
<tr><td colspan="2">Noticia de la semana.</td><td colspan="3" rowspan="2">CONTENIDOS</td></tr>
<tr><td colspan="2">Barra de links y aplicaciones.</td></tr>
<tr><td colspan="5">Información de copyrigh, y pie de página</td></tr>
</table>

Se define que el módulo php que mostrará la noticia de la semana se denominará noticia.php

Ejercicio: Modificar el archivo layout1.php para reflejar el cambio.

A continuación una tarea habitual es que el equipo de diseño realice un archivo html (o php) con un ejemplo de cómo debe quedar para que sea visualmente agradable el módulo noticia de la semana, por ejemplo puede verse de la siguiente manera:



El código HTML del módulo noticia.php es el siguiente:

```
<table bgcolor="#22AE22" width="140" border="0" cellspacing="0" cellpadding="0">
<tr><td>
<img src="flechita.gif">
<b>Título Noticia</b>
</td></tr>
<tr><td bgcolor="#226622">
Información sobre la noticia datos varios cosas divertidas.
</td>
</tr>
</table>
```

Lo que debemos hacer es reemplazar el título de la noticia y el texto de la misma por valores que obtenemos de la base de datos. Para ello decidimos crear una tabla llamada “noticia” en donde almacenaremos la noticia de la semana, además tenemos que construir un cargador que permita insertar el título y texto de la noticia en la base.

Definimos la tabla noticia en MySQL de la siguiente manera:

```
CREATE TABLE noticia(
  título varchar(40),
  texto text,
  fecha datetime
);
```

La fecha la vamos a usar para poder insertar varias noticias y que en la página se muestre siempre la última (la que tiene fecha más reciente), el historial de noticias de la semana puede usarse eventualmente en otra página para mostrar un listado de los títulos de la noticia de la semana, etc.

El formulario cargador que vamos a utilizar usa las técnicas que vimos en el capítulo de formularios y le agregamos el manejo de la base de datos para que realice el insert en la tabla de noticias.

```
<?
//Este código se ejecuta cuando se presiono el botón de submit
if(isset($proc)) {
 $dab=mysql_connect("localhost","root","seldon");
 mysql_select_db("curso",$dab);
 $query="INSERT into noticia(título,texto,fecha) values('$título','$texto',now()
)";
 $res=mysql_query($query,$dab);
 if(!$res) {
  $x=mysql_error();
  print("Se produjo un error al insertar: $x");
 }
}
?>
<form action="<?print("$PHP_SELF");?>" method="post">
Título de la noticia: <input type="text" maxlength="40" name="título"> <BR>
Texto de la noticia: <textarea rows="10" cols="80" name="texto"></textarea><BR>
<input type="submit" name="proc">
</form>
```

Luego podemos modificar el módulo noticia.php para que obtenga el título y texto de la última noticia y sea esto lo que se muestre al usar el módulo.

Noticia.php modificado queda de la forma:

```
<?
//Obtenemos los datos de la noticia a buscar
$dab=mysql_connect("localhost","root","seldon");
if(!$dab) {$título="No hay noticia";$texto="No hay noticia";} else {
  mysql_select_db("curso",$dab);
  $query="select título,texto from noticia order by fecha desc";
  $res=mysql_query($query,$dab);
  if(!$res) {$título="No hay noticia";$texto="No hay texto";} else {
   $r=mysql_fetch_row($res);
   $título=$r[0];
   $texto=$r[1];
  }
}
?>
<table bgcolor="#22AE22" width="140" border="0" cellspacing="0" cellpadding="0">
<tr><td>
<img src="flechita.gif">
<b><?print("$título");?></b>
</td></tr>
<tr><td bgcolor="#226622">
<?print("$texto");?>
</td>
</tr>
</table>
```

Como podemos ver probando el módulo o usando el layout modificado con la inclusión de este módulo a partir de este momento la noticia de la semana que se muestra al ingresar a la página es la última noticia ingresada en la base, usando el cargador de noticias puede modificarse la misma cargando una nueva noticia y la próxima vez que se acceda a la base automáticamente se ve la nueva noticia. Este es el principio básico de un web-site dinámico y de la generación dinámica y en tiempo real de páginas web.

Almacenamiento de objetos binarios en MySQL:

Un tema interesante en Mysql es usar la base de datos para guardar además de datos de tipo texto datos binarios como por ejemplo código html o imágenes. De esta forma se pueden almacenar completamente en la base de datos todos los elementos necesarios para construir un web-site.

```
CREATE TABLE binary_data (
id INT(4) NOT NULL AUTO_INCREMENT PRIMARY KEY,
description CHAR(50),
bin_data LONGBLOB,
filename CHAR(50),
filesize CHAR(50),
filetype CHAR(50)
);
```

El siguiente script puede usarse para insertar objetos binarios en la base de datos desde un browser. Notar que se usa el tag input type=”file” de un form html para subir un archivo.

```
<HTML>
<HEAD><TITLE>Store binary data into SQL Database</TITLE></HEAD>
<BODY>

<?php
if ($submit) {
//codigo que se ejecuta si se presiono el botón submit
MYSQL_CONNECT( "localhost", "root", "password");
mysql_select_db( "binary_data");
$data = addslashes(fread(fopen($form_data, "r"), filesize($form_data)));
$result=MYSQL_QUERY( "INSERT INTO binary_data
(description,bin_data,filename,filesize,filetype) ".
"VALUES
('$form_description','$data','$form_data_name','$form_data_size','$form_d
ata_type')");
$id= mysql_insert_id();
print "<p>Database ID: <b>$id</b>";
MYSQL_CLOSE();
} else {
// sino mostrar el formulario para nuevos datos:
?>
<form method="post" action=" <?php echo $PHP_SELF; ?>"
enctype="multipart/form-data">
File Description:<br>
<input type="text" name="form_description" size="40">
<INPUT TYPE="hidden" name="MAX_FILE_SIZE" value="1000000">
<br>File to upload/store in database:<br>
<input type="file" name="form_data" size="40">
<p><input type="submit" name="submit" value="submit">
</form>
<?php
}
?>
</BODY>
</HTML>
```

Notar el uso de la variable predefinida $PHP_SELF que tiene el nombre del script de esta forma el formulario se llama a si mismo independientemente del nombre que se le ponga al archivo.

El siguiente script (getdata.php) puede usarse para recuperar los datos desde la base de datos, notar que el script espera recibir la variable $id con el id del registro a recuperar de la tabla.

```
<?php
if($id) {
@MYSQL_CONNECT( "localhost", "root", "password");
@mysql_select_db( "binary_data");
$query = "select bin_data,filetype from binary_data where id=$id";
$result = @MYSQL_QUERY($query);
$data = @MYSQL_RESULT($result,0, "bin_data");
$type = @MYSQL_RESULT($result,0, "filetype");
Header( "Content-type: $type");
echo $data;
};
?>
```

Para usar una imagen que se obtiene de la base de datos se puede usar:

```
<img src="getdata.php?id=3">
```

Notar como se le pasa la variable id al script para saber cual es el registro a recuperar de la base.

Usando este conjunto de scripts es sencillo:

- Armar un cargador que permita insertar en la base de datos archivos binarios (imágenes, código html, etc)
- Utilizar los datos almacenados en la base para insertarlos en una página web usando el script getdata o una variante del mismo.

Capitulo 6: Manejo de Archivos.

1. Apertura de un archivo.

La función utilizada para abrir un archivo en PHP es fopen, la sintaxis.

fp_handler=fopen("path","modo");

Parh es la ruta completa del archivo a abrir, si el path comienza con "http://" se realiza una conexión a la URL indicada y se abre la página como si fuera un archivo (con las limitaciones lógicas, por ejemplo no es posible escribir).

Los modos en los que se puede abrir un archivo son:

r	Sólo lectura
r+	Lectura y escritura
w	Sólo escritura, si no existe el archivo lo crea, si existe lo trunca
w+	Lectura y escritura, si existe lo trunca, si no existe lo crea
a	Modo append sólo escritura si no existe lo crea
a+	Modo append lectura y escritura si no existe lo crea

La función devuelve un file_handler que luego debe ser usado en todas las funciones de tipo fnombre_funcion, como por ejemplo fgets, fputs, fclose, fread, fwrite, etc.

2. Lectura desde un archivo.

Las funciones que pueden usarse para leer un archivo son:

string=**fgets**(file_handler, longitud)

Lee una línea de texto hasta el fin de línea o bien hasta que se cumpla la longitud indicada, devuelve el resultado en la variable pasada. El archivo debe estar abierto con fopen.

var=**fread**(file_handler, cantidad)

Lee la cantidad de bytes indicados ignorando saltos de línea y deja el resultado en la variable var.

Ejemplo

$buffer=**fread**($fp, 1024); //Lee 1Kb desde el archivo cuyo handler es $fp

string=**fgetss**(file_handler, longitud)

Idéntica a fgets con la diferencia de que los tags html son eliminados del archivo a medida que se lee el mismo. Opcionalmente puede pasarse una lista de tags que no deben ser eliminados.

Ejemplo:

$string=fgetss($fp,999999,"<b> <i> <table> <tr> <td>");

Lee una línea (de cualquier longitud) eliminando los tags html excepto los indicados como segundo parámetro. Los tags que cierran los tags especificados en la lista de tags permitidos tampoco son eliminados.

3. Escritura a un archivo

fwrite(file_handler, variable, longitud);

Escribe la variable al archivo indicado por file_handler. Si esta indicado el parámetro "longitud" (que es opcional) se escribirán tantos bytes como la longitud indicada por dicho parámetro o como la longitud de la variable, en aquellos casos en que el parámetro longitud es mayor que la longitud de la variable.

La función devuelve la cantidad de bytes escritos en el archivo.

Ejemplo:

```
$q = fwrite($fp,$buffer,999999);
```

fputs es idéntico a fwrite y funciona de la misma manera. (es un alias).

4. Cierre de archivos

fclose(file_handler)

Cierra un archivo abierto con fopen.

5. Fin de archivo

boolean = feof(file_handler);

Devuelve verdadero si no quedan más bytes para leer en el archivo o si se produce algún tipo de error al leer.

Ejemplo:

```
$fp=fopen("/usr/luis/archivo.txt","r");
while(!feof($fp)) {
  $s=fgets($fp,999999);
  print("$s");
}
```

6. Manejo de archivos.

PHP provee funciones para copiar, renombrar, mover y borrar archivos y directorios, las funciones son:

Función	Descripción
rename(path_origen, path_destino);	Renombra un archivo. Ejemplo: $newname="/usr/eduardo/file.txt"; Rename("/usr/eduardo/archivo.txt","$newname");
unlink(path_a_borrar);	Elimina un archivo.
rmdir(directorio_a_borrar);	Elimina un directorio (debe estar vacío)
mkdir(path_a_crear);	Crea un directorio Nuevo.
copy(path_origen, path_destino);	Copia un archivo o varios.

7. Otras funciones útiles.

Función	Descripción
fseek(file_handler,posicion,desde)	Posiciona el puntero de lectura/escritura de un archivo en el offset indicado. El parámetro "desde" es opcional y puede tomar uno de los siguientes valores: SEEK_SET (el offset es absoluto) SEEK_CUR (el offset es relativo a la posición actual) SEEK_END (el offset es desde el final del archivo) El default es SEEK_SET
ftruncate(file_handler, longitud)	Trunca el archivo indicado a la longitud en bytes especificada.
array=file(path)	Lee un archivo de texto y devuelve un vector donde cada elemento del vector es una línea del archivo.
file_exists(path)	Devuelve true/false según el path indicado exista o no.
filemtime(path)	Devuelve la fecha de última modificación de un archivo en formato Unix. (ver manejo de fechas)
filesize(path)	Devuelve el tamaño de un archivo.
filetype(path)	Devuelve el tipo de un archivo.
flock(file_handler,modo)	Lockea un archivo (independientemente del file-system), el modo puede ser: 1: Lock en modo lectura (compartido) 2: Lock en modo escritura (exclusivo) 3: Release del lock adquirido. Al hacer un fclose del archivo se liberan automáticamente los locks adquiridos sobre el mismo. Si flock no puede obtener el lock espera hasta que el lock este disponible, si se quiere que flock no bloquee el script sumar 4 al modo (modos: 5,6,7) y consultar por el valor devuelto por la función: true si el lock fue adquirido o false si no fue adquirido. Usando esta función pueden implementarse mecanismos de sincronización entre procesos.
fpassthru(file_handler)	Escribe al standard_output el contenido del archivo.
readfile(path)	Lee todo el archivo y lo escribe al standard output.
is_dir(path)	True/false según el path sea un directorio
is_file(path)	True/false según el path sea un archivo
tempnam(path)	Dado el nombre de un directorio devuelve un nombre de archivo que no existe en el directorio dado (útil para crear archivos temporarios)

Manejo de directorios.

Las siguientes funciones de PHP están orientadas a recorrer directorios y obtener los archivos que se encuentran dentro de ellos. Las funciones son independientes del file-system con lo cual funcionan correctamente tanto en UNIX como en Windows.

directory_handle = opendir(path);

Abre el directorio pasado como parámetro y devuelve un handler al directorio, previamente puede usarse la función is_dir(path) para verificar si el path es un directorio en caso de que esta validación sea necesaria.

string = readdir(directory_handler)

Lee la próxima entrada en el directorio abierto con opendir, usualmente las dos primeras entradas de un directorio con "." y ".." por lo que el código que procesa los archivos del directorio debe tener esto en cuenta para no procesar dichas entradas en caso de que tal cosa no sea deseable.

closedir(directory_handler)

Cierra un handler abierto con opendir.

rewinddir(directory_handler)

Rebobina el puntero interno de un directorio para que apunte nuevamente al comienzo del mismo.

Capitulo 7: Programación Orientada a Objetos en PHP

Para comenzar a hablar de programación orientada a objetos (OOP – Object Oriented Programming) es necesario recordar los conceptos de la programación orientada a objetos. Estos conceptos varían entre los distintos autores, pero podemos mencionar algunos que son básicos y necesarios para cualquier lenguaje del cual pueda decirse que es orientado a objetos:

- Tipos de datos abstractos e información encapsulada
- Herencia
- Polimorfismo

La encapsulación en PHP se codifica utilizando clases:

```
<?php

class Algo {
// En OOP se utilizan generalmente nombres comenzados con mayúscula.
var $x;

function setX($v) {
// Para los métodos se utilizan generalmente nombres en minúscula y sólo
// se utiliza mayúscula para separar las palabras, por ej. getValueOfArea()
$this->x=$v;
}

function getX() {
return $this->x;
}
}

?>
```

Obviamente está nomenclatura es sólo a valor de recomendación para mantener un standard entre el código de los distintos programadores, y puede no ser respetado. Lo importante es acordar una nomenclatura standard que todos respeten.

Las propiedades de los objetos son definidas en PHP utilizando la declaración “var” dentro de la clase. Cuando se declara una propiedad la misma no tiene tipo alguno asignado, hasta que nosotros la asignemos algún valor en particular. Una propiedad puede ser un entero, un vector, un vector asociativo, e inclusive puede ser otro objeto.

Los métodos son definidos como funciones, también dentro de la clase,. Y para acceder a las propiedades de la instancia de esa clase es necesario referirse a las propiedades como $this->name. En caso de no utilizar el “$this->” la variable será local al método y una vez terminada la ejecución del mismo se perderá su valor.

Para crear una instancia de un objeto debemos ejecutar el operador “new”, que nos devuelve en una variable un objeto de la clase que le estamos indicando.

$obj = new Something;

Una vez instanciado el objeto podemos utilizar sus métodos:

$obj->setX(5);
$see=$obj->getX();

El método setX ejecutado en el objeto $obj hizo que se asigne un 5 a la propiedad “x” de dicha instancia. Notemos en este punto que podríamos haber seteado la propiedad “x” con cualquier tipo de variables, por ejemplo un string.

Para asignarle 5 a la propiedad “x” de nuestro objeto $obj podríamos haber puesto en nuestro código directamente “$obj->x=5;”, sin la necesidad de llamar a ningún método, pero el problema radicaría en que en que estaríamos violando la regla de encapsulamiento de los objetos. Una buena práctica de la programación orientada a objetos es acceder a las propiedades solamente mediante métodos propios de la clase y jamás acceder a ellos de otra forma. Lamentablemente PHP no ofrece la posibilidad de declarar las propiedades privadas, por lo que el programar en forma “encapsulada” se torna más una filosofia de programación que una obligación.

La herencia en PHP se realiza utilizando la sentencia “extends”:

```
<?php

class Another extends Something {
var $y;
function setY($v) {
// Para los métodos se utilizan generalmente nombres en minúscula y sólo
// se utiliza mayúscula para separar las palabras, por ej. getValueOfArea()
$this->y=$v;
}

function getY() {
return $this->y;
}
}

?>
```

Los objetos de la clase “Another” poseen todas las propiedades y métodos de su clase padre “Something”, más las propiedades y métodos propios. Ahora podemos ejecutar por ejemplo los siguientes comandos:

$obj2=new Another;
$obj2->setX(6);
$obj2->setY(7);

En PHP una única clase de objetos no puede ser “hija” de más de un “padre”, lo que es conocido como múltiple herencia.

En PHP se pueden reescribir métodos de la clase padre en la clase hijo (overriding). Para esto sólo hace falta volver a definir la función en el objeto hijo. Por ejemplo si queremos redefinir el método getX para la clase “Another” simplemente definimos la función en la clase “Another”. Una vez hecho esto no es posible para los objetos de la clase “Another” acceder al método getX de “Something”.

En caso de declararse una propiedad en la clase “hija” con el mismo nombre que en la clase padre, la propiedad de la clase padre se encontraría “oculta”.

En las clases se pueden definir constructores. Los constructores son métodos que se ejecutan al momento de instanciar la clase (cuando se ejecuta la sentencia new). La característica para que un método sea el constructor de una clase es que debe tener el mismo nombre que la clase.

```
<?php

class Something {
```

```
var $x;

function Something($y) {
$this->x=$y;
}

function setX($v) {
$this->x=$v;
}

function getX() {
return $this->x;
}
}

?>
```

Entonces, se puede crear el objeto de la siguiente manera:

$obj=new Something(6);

Automáticamente el constructor asigna 6 a la propiedad “x”.

Todos los métodos, incluyendo los constructores, son funciones normales de php, por lo que se le pueden asignar valores por omisión.
Supongamos que tenemos el constructor definido de la siguiente manera:

function Something($x="3",$y="5")

En este caso podemos crear el objeto de la siguiente manera:

$obj = new Something(); // x=3 y y=5
$obj = new Something(8); // x=8 y y=5
$obj = new Something(8,9); // x=8 y y=9

Los argumentos por omisión son utilizados en C++ y son una vía para cuando no hay valores para pasar por parámetro a las funciones. Cuando el parámetro no es encontrado por la función que es llamada, toma el valor por omisión que le es especificada en su definición.

Cuando es creado un objeto de una clase que deriva de otra, se ejecuta sólo el constructor de la misma clase, y no la del padre. Este es un defecto del PHP, ya que es clásico en los lenguajes orientados a objetos que exista lo que se llama encadenamiento de constructores. Para hacer esto en PHP es necesario llamar explícitamente el constructor de la clase padre dentro del constructor de la clase heredera. Esto funciona porque todos los métodos de la clase padre se encuentran disponibles en la clase heredera.

```
<?php

function Another() {
$this->y=5;
$this->Something(); //Llamada explícita al constructor de la clase padre.
}

?>
```

Un buen mecanismo en OOP es usar clases abstractas. Clases abstractas son aquellas clases que no son instanciables y están hechas para el único propósito de definir una interface para otras clases derivadas. Los diseñadores

usualmente utilizan estas clases para forzar a los programadores a derivar clases desde ciertas bases clases de forma tal de asegurarse que cada clase tiene una cierta funcionalidad predefinida. No hay una forma standard de hacer esto en PHP pero: Si se necesita esta metodologia puede definirse una clase base y poner una sentencia "die" en el constructor, de esta forma nos aseguramos que la clase no es instanciable luego definimos los metodos interface poniendo una sentencia "die" en cada uno de ellos, de esta forma los programadores deben sobreescribir estos metodos para poder utilizarlos.

No hay destructores en PHP.

La sobrecarga (overloading) de metodos que es diferente a la redefinicion (overriding) no esta soportada en PHP. En OOP se dice que un metodo esta sobrecargado cuando hay mas de un metodo con el miusmo nombre pero diferentes tipos o cantidad de parametros. PHP es un lenguaje debilmente tipado por lo que la sobrecarga por tipos no funcionaria, por consistencia la sobrecarga por cantidad de parametros tampoco funciona.

En OOP es agradable muchas veces sobrecargar constructores de forma tal que una clase permita construir un objeto de muchas formas diferentes, podemos simular esto en PHP de la forma:

```
<?php

class Myclass {
function Myclass() {
$name= "Myclass".func_num_args();
$this->$name();
//Notar que $this->$name() es susualmente erroneo porque aquí
//$name es un string con el nombre del método a llamar.
}

function Myclass1($x) {
code;
}
function Myclass2($x,$y) {
code;
}
}

?>
```

Con éste trabajo extra podemos trabajar con la clase de un modo transparente para el usuario:

$obj1=new Myclass('1'); //Will call Myclass1
$obj2=new Myclass('1','2'); //Will call Myclass2

Polimorfismo

Polimorfismo se define como la habilidad de un objeto de determinar que método debe invocar para un objeto pasado como argumento en tiempo de ejecución. Por ejemplo si tenemos una clase figura que tiene un método "dibujar" y sus clases derivadas circulo y rectángulo, cuando reemplazamos el método "dibujar" podemos tener una función que cuente con un argumento "x" y después llamar a $x->dibujar(). Si tenemos polimorfismo el método "dibujar" llamado dependerá del tipo de objeto que pasemos a la función. El polimorfismo es muy fácil de aplicar y utilizar en lenguajes interpretados como PHP, pero no es tan secillo en lenguajes compilados, ya que no sabemos que método deberemos llamar de antemano.

```
<?php
```

```php
function niceDrawing($x) {
//Supongamos que este es un método de la clase Board.
$x->draw();
}

$obj=new Circle(3,187);
$obj2=new Rectangle(4,5);

$board->niceDrawing($obj); //Podemos llamar al método draw de circulo.
$board->niceDrawing($obj2); //Podemos llamar al método draw de rectángulo.

?>
```

Programación Orientada a Objetos en PHP

Algunos puristas pueden decir que PHP no es un verdadero lenguaje orientado a objetos. PHP es un lenguaje hibrido donde se puede utilizar OOP o procedimientos de programación procedural tradicional. Para largos proyectos usted pude querer o necesitar usar OOP "puro" en PHP declarando clases y utilizando sólo objetos y clases para su proyecto. Mientras más largo es el proyecto más puede ayudar la programación orientada a objetos, ya que código es más fácil de mantener, entender y reutilizar. Estos son los principios básicos de la Ingeniería de Software- Aplicar estos conceptos a proyectos basados en la web será la llave de acceso para exitosos sites en el futuro.

Técnicas avanzadas de programación orientada a objetos.

Luego de haber visto los conceptos básicos de la OOP, vamos a ver algunas técnicas más avanzadas:

Serialización

PHP no soporta la persistencia de objetos. En OOP los objetos persistentes son objetos que mantienen su estado y funcionalidad a través de múltiples invocaciones de la aplicación. Esto puede resolverse salvando el objeto en un archivo o en una base de datos y restaurando los datos cada vez que se ejecuta dicha aplicación. El mecanismo es conocido como serialización. PHP provee un método de serialización que puede ser llamado por los objetos . El método de serialización devuelve un string representando el objeto. La serialización guarda las propiedades del objeto pero no sus métodos.

En PHP4 si se serializa un objeto al string $s, se destruye el objeto, y entonces utilizando la desserialización de objeto en $obj se puede mantener el acceso a las propiedades del objeto. Esto no es recomendado por dos razones: la primera es porque no se garantiza que en futuras versiones esto siga funcionando. La segunda es porque si se serializa un objeto, se guarda a disco el string y se sale del script, al correr en el futuro dicho script no nos aseguramos que los métodos del objeto sean los mismos, ya que en la serialización sólo se guardaron las propiedades. Concluyendo, serialización sólo sirve en PHP para guardar las propiedades de un objeto nada más (se puede serializar un vector asociativo para salvarlos a disco por ejemplo).

NOTA: La version 4.0.1 de PHP4 soporta serializacion de objetos en forma completa!!!

Ejemplo: <?php

```php
$obj=new Classfoo();
$str=serialize($obj);
// Se salva $str al disco
```

```
//...algunos meses más tarde

//Se recupera str del disco
$obj2=unserialize($str)

?>
```

En este caso tenemos las propiedades recuperadas pero no podemos utilizar los métodos. Esto hace que la forma de recuperar $obj2->x el valor de “x” sea la única forma posible (no tenemos métodos!), por lo que esta práctica no es recomendada.

Estas son algunas maneras para arreglar el problema, las que veremos muy por encima, ya que son formas muy enmarañadas de resolverlo. Full serialización será uno de los agregados más esperados en PHP.

Nota: Enhorabuena, PHP 4.0.1 soporta “full serialization” de objetos.

Usando clases para manipular información almacenada

Una de las cosas útiles de PHP y OOP que se pueden definir fácilmente clases para manejar ciertas cosas y llamar a la clase apropiada cuando queramos. Supongamos que tenemos un formulario html donde un usuario selecciona un producto por medio de su producto ID. Supongamos también que tenemos los datos del producto en una base de datos y queremos mostrar el producto en sí, el precio y demás características. Si tenemos productos de diferentes tipos, y en la misma acción tenemos diferentes maneras para diferentes familias de productos. Por ejemplo para algunos porductos debemos reproducir un sonido mientras que para otros productos debemos mostrar una foto del mismo guardada en la base de datos. En este caso podemos utilizar OOP y PHP para realizar menos y mejor código fuente:

Podemos definir una clase Producto, definimos que métodos la clase debe tener (por ejemplo “display”), y entonces definimos clases para cada tipo de producto como una extensión (extends) la clase producto (por ejemplo clase ItemSonoro,. ItemVisible, etc.) redefiniendo los métodos que ya definimos en Producto en cada una de las clases que necesitemos. Lo que resulta conveniente en este caso es nombrara las clases de acuerdo con los tipos de “columna” que guardamos en la base de datos por cada tipo de producto que tenemos (id, tipo, precio, etc.). Entonces cuando procesamos el script podemos pedir el tipo desde la base de datos e instanciar un objeto de la clase del tipo de producto:

```
<?php

$obj=new $type();  // type has the name of the class to instanciate!
$obj->action();

?>
```

Esto es una buena propiedad de PHP que permite llamar al método “display” por ejemplo del tipo de objeto que tenemos. Con esta técnica no necesitamos tocar el script de proceso (el formulario) cuando agregamos un nuevo tipo de objetos, solamente agregamos la clase y listo. Esto es poderoso, lo único qu debemos definir son los métodos que todos los objetso deben tener de acuerdo a los tipos que tenemos, generando las diferentes maneras para los diferentes tipos.

Si ud. Lidera un grupo de programandores es fácil dividir tareas, cada uno responsable por cada tipo de objetos y las clases que dependen de él. Con estos métodos la internacionalización de un site puede ser muy fácil aplicando los métodos correspondientes de acuerdo al lenguaje que seleccionó el usuario.

Copiando y Clonando

Cuando creamos un objeto $obj se puede copiar un objeto usando $obj2=$obj. El nuevo objeto es una copia del objeto $obj, es decir que tiene el estado que $obj tenia en el momento que se realizó la asignación. Algunas veces no necesitamos esto, sólo queremos crear un nuevo objeto de la misma clase que $obj, llamando al contructor en el momento de la creación del nuevo objeto. Esto es posible utilizando la serialización y una clase que todas las otras clases sean extensión de la misma.

Entrando a una zona difícil

Cuando serializamos un objeto obtenemos un string de un formato propio. Podemos investigas (siendo curiosos) cada una de las cosas que tiene dicho string. Una cosa curiosa es que esta guardado en el el nombre de la clase que debemos utilizar para desserializar el objeto:

```
<?php

$herring=serialize($obj);
$vec=explode( ':',$herring);
$nam=str_replace( "\"", '',$vec[2]);

?>
```

Entonces supongamos que creamos una clase “Universe” y forzamos que todas las clases sean extensión de universo, podemos definir un método que clone Universe como:

```
<?php

class Universe {
function clone() {
$herring=serialize($this);
$vec=explode( ':',$herring);
$nam=str_replace( "\"", '',$vec[2]);
$ret=new $nam;
return $ret;
}
}

//Entonces:

$obj=new Something();
//Algo extensión de universo !!
$other=$obj->clone();

?>
```

Se obtiene un objeto nuevo de la clase “Something” creado de la misma forma que llamando a New, se invoca al constructor etc. Distintas aplicaciones del uso de la clase universal permiten varios manejos curiosos de objetos en PHP, el unico limite es la imaginacion!

Capítulo 9: Persistencia.

Uno de los problemas clásicos en el desarrollo de web sites y aplicaciones web es la perdida de persistencia cuando el usuario pasa de una página a otra. Debido a las características de diseño del protocolo HTTP que fuerza una nueva conexión y desconexión por cada request no es posible saber quien está accediendo a que página o en que lugar esta cada usuario del site. Mantener persistencia a lo largo de la navegación del sitio ha sido uno de los temas más complejos e importantes en el desarrollo de aplicaciones web, en este capítulo ilustraremos distintos métodos que pueden usarse en PHP para mantener persistencia.

Sesiones:

Se suele definir como una sesión al tiempo en el que un usuario determinado se encuentra navegando en el site, dependiendo de la definición podemos decir que si el usuario no navega por el site durante una cierta cantidad de minutos ha terminado su sesión en el sitio y a partir de allí cuando vuelve a ingresar lo hace en una nueva sesión. El concepto de sesión es útil porque es posible asociar a cada sesión un identificador único de forma tal de registrar la actividad del usuario en el site y mantener persistencia utilizando únicamente este identificador, el problema pasa a ser como mantener la persistencia del identificador de sesión (SID) de ahora en adelante, y las posibilidades son las que detallamos a continuación:

1. Cookies

Uno de los mecanismos más usados para mantener persistencia es el mecanismo de cookies, inventado por Netscape y hoy en día aceptado por casi todos los browsers, en especial los más populares. El concepto es que mediante un header del protocolo HTTP el server pueda almacenar información en el cliente. A esta información que el server guarda en el cliente se la denomina “cookie”. Las cookies pueden habilitarse o deshabilitarse desde el browser por lo que algunos usuarios no lo soportan, son de uso bastante general en muchos web sites a punto tal que en sites de la importancia de yahoo si el usuario no tiene habilitadas las cookies prácticamente no puede utilizar la mayoría de los servicios del site. Cuando el server envía un header con un cookie el browser, si acepta cookies, guarda la información enviada en un archivo de texto con un formato especial. Cada vez que el browser solicita una página del dominio que envió la cookie re-envia la cookie al site, de esta forma es posible mantener persistencia. La información que puede guardarse en una cookie esta limitada por lo que habitualmente se utiliza la misma para mantener el identificador de sesión del usuario almacenándose el resto de los datos necesarios en el servidor usando el session-id de la cookie como clave.

Para crear un cookie en PHP se utiliza la función setcookie cuya sintaxis es la siguiente:

int=setcookie(nombre, valor, expiración, path, dominio);

Nombre	: Nombre de la cookie a setear por ejemplo “sesion”
Valor	: Valor que contendrá la cookie, como por ejemplo “khdhkfdh47”
Expiracion	: Fecha de vencimiento de la cookie (fecha en la cual el browser la borra del disco del usuario), debe estar en formato Unix. En general el uso más practico es time()+tiempo donde tiempo es la cantidad de segundos de vida de la cookie.
Path	: En general no se usa, suele setearse en “/”
Dominio	: Dominio para el cual el cookie es valido ejemplo “.prueba.com” en cuyo caso sirve para algo.prueba.com, site1.prueba.com, site2.prueba.com y todos los de la misma forma.

La función devuelve verdadero si pudo setearse la cookie o falso en caso contrario (por ejemplo si el browser no acepta cookies)

Ejemplo:

$val=setcookie(“sesion”,”1”,time()+3600,”/”,”.prueba.com”);

Para recuperar el valor de una cookie se debe usar el vector de PHP $HTTP_COOKIE_VARS que es u vector asociativo indexado por nombre de cookie.

Ejemplo:

$ck=$HTTP_COOKIE_VARS[“sesion”];

Con lo cual se recupera el valor de la cookie.

Las cookies son sumamente practicas para manejar sesiones, en cada página se verifica si el usuario tiene seteada la cookie con nombre “session” si la tiene se recupera el valor de la sesión, si no la tiene se crea un identificador de sesión nuevo y único y se setea la cookie correspondiente con el vencimiento que corresponde según la duración que uno considere necesaria. Luego el identificador de sesión es accesible en cada página para almacenar valores por ejemplo en la base de datos como el nombre del usuario, preferencias de la sesión y otros valores. La recuperación de la cookie y su creación en caso de no existir se puede colocar en un modulo php que luego se incluye en cada página por ejemplo mod_session.php solo hay que recordar en cada página hacer un include(“mod_session.php”); y luego se puede usar la variable $session_id (o el nombre que se le haya dado en el modulo) para guardar y recuperar valores correspondientes a la sesión actual.

2. URL

Otro método posible para pasar información desde una página a otra es mediante el URL de la página en cuestión, usando el URL se pueden pasar datos de una página a otra usando el query string de la forma:

http://dominio/path?query_string

Donde query_string es de la forma: variable=valor&variable2=valor2&variable3=valor3 … etc…

De esta forma podríamos hacer un manejo similar al anterior pero pasando el session_id usando el url en lugar de usando cookies, la desventaja de este método es que todos los links deben generarse dinámicamente en PHP para agregar a la dirección del link el valor del cookie de la forma:

<a href=”http://dominio/path?session=<?print(“$session_id”);?>”>

El funcionamiento es similar, no requiere que el browser tenga habilitados cookies pero altera la forma en que se escriben los links y afea un poco la forma en la cual se muestra la URL de la página actual, podría por ejemplo tener consecuencias como entorpecer la tarea de generar el bookmark de una determinada página.

3. Sesiones en PHP

PHP soporta en forma nativa desde el lenguaje el concepto de sesiones, en PHP se pueden manejar sesiones en forma transparente al usuario, el lenguaje define una constante PHPSESSID con el identificador de cada sesión y se encarga de propagar el mismo usando cookies o bien el URL de la página en caso de que los cookies estén deshabilitados.

Para usar cookies en PHP es necesario invocar la función session_start(); en el comienzo del script, esta función se encarga de recuperar el session_id en la constante $PHPSESSID y si la sesión no existe crea una nueva.

Si PHP esta compilado con la opción –enable-trans-sid el interprete se encarga de re-escribir dinámicamente las URLs agregando el session_id en caso de que el usuario no tenga cookies, en caso contrario hay que modificar los links de la forma:

<a href=”/lugar?PHPSESSID=<?echo “$PHPSESSID”?>”>

PHP dispone además de funciones para “registrar” variables dentro de una sesión, esto se hace llamando a la función session_register(); Por ejemplo:

```
<?
session_start();
$x=”hola”;
session_register(“x”);
?>
```

Notar que session_register recibe el “nombre” de la variable a registrar sin “$”. Una vez registrada una variable la misma estará disponible en todas las páginas que usen session_start durante la sesión actual, es decir que si desde la página donde hicimos el session_register el usuario se traslada a otra página que usa session_start entonces la variable $x automáticamente estará definida y con el valor “hola” ya que es una variable registrada dentro de la sesión. Cualquier tipo de variable de PHP puede registrarse con este método, incluyendo objetos (sólo las propiedades del objeto se registran, no los métodos!)

Otras funciones de manejo de sesiones:

Función	Descripción
session_id();	Devuelve el id de la sesión actual, si se le pasa como parámetro un string cambia el id de la sesión al string pasado, esta función hace lo mismo que la consulta directa a la variable $PHPSESSID
session_destroy();	Destruye la sesión actual y todas las variables registradas.
session_name();	Devuelve el nombre de la sesión actual (PHPSESSID), si se le pasa como parámetro un string cambia el nombre de la sesión y por consiguiente de la variable que contiene el id de la sesión al valor pasado.
session_unregister(variable_name);	Elimina de la sesión una variable registrada con session_register
session_is_registered(variable_name);	Dice si una variable esta o no registrada en la sesión actual (devuelve true/false), notar que es equivalente a la función isset pero tomando en cuenta la sesión.

Manejo avanzado de sesiones en PHP.

Una pregunta interesante relativa al manejo de sesiones en PHP es como se almacenan los datos de la sesión (las variables registradas y demás), en PHP por omisión los datos sobre las sesiones se almacenan en archivos en el directorio /tmp, este método funciona correctamente pero tiene 2 inconvenientes:

1. No es fácil compartir sesiones entre servers.

   Dado que los datos se almacenan en un file-system una variable “registrada” en un server no puede ser visible en otro ya que el mismo no tiene forma de recuperar el contenido de la misma. (Podría hacerse si se cambia el directorio /tmp a un directorio compartido entre maquinas usando NFS pero este no es el tema aquí)

2. Dada una cantidad de usuarios realmente elevada el método no es eficiente.

Estamos hablando de cantidades de usuarios realmente muy grandes simultáneos, dados los cuales hay demasiadas sesiones por mantener y el file-system se torna lento en la búsqueda y recuperación de los archivos con los datos.

Afortunadamente en PHP es posible redefinir los métodos de almacenamiento que el lenguaje utiliza para almacenar las sesiones. Esto se hace con la función session_set_save_handler

Ejemplo:
session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");

Que recibe como parámetros los "nombres" de 6(seis) funciones, cuyos prototipos y funcionalidad son los siguientes:

Función	Descripción
function open ($save_path, $session_name)	Se llama luego de cada session_start, recibe un path y un nombre de sesión que por omisión es PHPSESSID.
function close()	Cierra la sesión.
function read ($key)	Dada una clave "key" recupera para la sesión actual el valor de la clave pasada.
function write ($key, $val)	Guarda un par clave-valor para la sesión actual.
function destroy ($key)	Destruye la clave pasada como parámetro.
Function gc($maxlifetime)	Garbage collection, se encarga de destruir los datos de las sesiones vencidas.

En general en open no es necesario hacer nada ya que todo el manejo de Registración de variables esta dado por read y write. Write recibirá como clave el identificador de la sesión y como datos una estructura interna de php que contiene la representación de todos los datos a almacenar para la sesión. De la misma forma read debe ,dada la clave (session_id), recuperar todos los datos de la sesión y luego php en forma interna setea las variables correspondientes para que sean visibles.

Todas las funciones deben devolver "true" para que el manejo de sesiones funcione.

A continuación dos módulos de Zing Yiang ilustrando como hacer un manejador de sesiones usando DBM y otro usando MySQL y un ejemplo de test.

```
<?
/* ------------------------------------------------------------------------
 * session_dbm.php
 * ------------------------------------------------------------------------
 * PHP4 DBM Session Handler
 * Version 1.00
 * by Ying Zhang (ying@zippydesign.com)
 * Last Modified: May 21 2000
 *
 * ------------------------------------------------------------------------
 * TERMS OF USAGE:
 * ------------------------------------------------------------------------
 * You are free to use this library in any way you want, no warranties are
 * expressed or implied.  This works for me, but I don't guarantee that it
 * works for you, USE AT YOUR OWN RISK.
 *
 * While not required to do so, I would appreciate it if you would retain
 * this header information.  If you make any modifications or improvements,
 * please send them via email to Ying Zhang <ying@zippydesign.com>.
 *
 * ------------------------------------------------------------------------
 * DESCRIPTION:
```

```
 * ------------------------------------------------------------------------
 * This library tells the PHP4 session handler to write to a DBM file
 * instead of creating individual files for each session.
 *
 * ------------------------------------------------------------------------
 * INSTALLATION:
 * ------------------------------------------------------------------------
 * Make sure you have DBM support compiled into PHP4.  Then copy this
 * script to a directory that is accessible by the rest of your PHP
 * scripts.
 *
 * ------------------------------------------------------------------------
 * USAGE:
 * ------------------------------------------------------------------------
 * Include this file in your scripts before you call session_start(), you
 * don't have to do anything special after that.
 */

$SESS_DBM = "";
$SESS_LIFE = get_cfg_var("session.gc_maxlifetime");

function sess_open($save_path, $session_name) {
       global $SESS_DBM;

       $SESS_DBM = dbmopen("$save_path/$session_name", "c");
       return ($SESS_DBM);
}

function sess_close() {
       global $SESS_DBM;

       dbmclose($SESS_DBM);
       return true;
}

function sess_read($key) {
       global $SESS_DBM, $SESS_LIFE;

       $var = "";
       if ($tmp = dbmfetch($SESS_DBM, $key)) {
              $expires_at = substr($tmp, 0, strpos($tmp, "|"));

              if ($expires_at > time()) {
                     $var = substr($tmp, strpos($tmp, "|") + 1);
              }
       }

       return $var;
}

function sess_write($key, $val) {
       global $SESS_DBM, $SESS_LIFE;

       dbmreplace($SESS_DBM, $key, time() + $SESS_LIFE . "|" . $val);
       return true;
}

function sess_destroy($key) {
       global $SESS_DBM;

       dbmdelete($SESS_DBM, $key);
       return true;
}
```

```
function sess_gc($maxlifetime) {
	global $SESS_DBM;

	$now = time();
	$key = dbmfirstkey($SESS_DBM);
	while ($key) {
		if ($tmp = dbmfetch($SESS_DBM, $key)) {
			$expires_at = substr($tmp, 0, strpos($tmp, "|"));
			if ($now > $expires_at) {
				sess_destroy($key);
			}
		}

		$key = dbmnextkey($SESS_DBM, $key);
	}
}

session_set_save_handler(
	"sess_open",
	"sess_close",
	"sess_read",
	"sess_write",
	"sess_destroy",
	"sess_gc");
?>


<?
/* ------------------------------------------------------------------------
 * session_mysql.php
 * ------------------------------------------------------------------------
 * PHP4 MySQL Session Handler
 * Version 1.00
 * by Ying Zhang (ying@zippydesign.com)
 * Last Modified: May 21 2000
 *
 * ------------------------------------------------------------------------
 * TERMS OF USAGE:
 * ------------------------------------------------------------------------
 * You are free to use this library in any way you want, no warranties are
 * expressed or implied.  This works for me, but I don't guarantee that it
 * works for you, USE AT YOUR OWN RISK.
 *
 * While not required to do so, I would appreciate it if you would retain
 * this header information.  If you make any modifications or improvements,
 * please send them via email to Ying Zhang <ying@zippydesign.com>.
 *
 * ------------------------------------------------------------------------
 * DESCRIPTION:
 * ------------------------------------------------------------------------
 * This library tells the PHP4 session handler to write to a MySQL database
 * instead of creating individual files for each session.
 *
 * Create a new database in MySQL called "sessions" like so:
 *
 * CREATE TABLE sessions (
 *      sesskey char(32) not null,
 *      expiry int(11) unsigned not null,
 *      value text not null,
 *      PRIMARY KEY (sesskey)
 * );
 *
```

```
 * ------------------------------------------------------------------------
 * INSTALLATION:
 * ------------------------------------------------------------------------
 * Make sure you have MySQL support compiled into PHP4.  Then copy this
 * script to a directory that is accessible by the rest of your PHP
 * scripts.
 *
 * ------------------------------------------------------------------------
 * USAGE:
 * ------------------------------------------------------------------------
 * Include this file in your scripts before you call session_start(), you
 * don't have to do anything special after that.
 */

$SESS_DBHOST = "localhost";                  /* database server hostname */
$SESS_DBNAME = "sessions";             /* database name */
$SESS_DBUSER = "phpsession";           /* database user */
$SESS_DBPASS = "phpsession";           /* database password */

$SESS_DBH = "";
$SESS_LIFE = get_cfg_var("session.gc_maxlifetime");

function sess_open($save_path, $session_name) {
       global $SESS_DBHOST, $SESS_DBNAME, $SESS_DBUSER, $SESS_DBPASS, $SESS_DBH;

       if (! $SESS_DBH = mysql_pconnect($SESS_DBHOST, $SESS_DBUSER, $SESS_DBPASS))
{
              echo "<li>Can't connect to $SESS_DBHOST as $SESS_DBUSER";
              echo "<li>MySQL Error: ", mysql_error();
              die;
       }

       if (! mysql_select_db($SESS_DBNAME, $SESS_DBH)) {
              echo "<li>Unable to select database $SESS_DBNAME";
              die;
       }

       return true;
}

function sess_close() {
       return true;
}

function sess_read($key) {
       global $SESS_DBH, $SESS_LIFE;

       $qry = "SELECT value FROM sessions WHERE sesskey = '$key' AND expiry > " .
time();
       $qid = mysql_query($qry, $SESS_DBH);

       if (list($value) = mysql_fetch_row($qid)) {
              return $value;
       }

       return false;
}

function sess_write($key, $val) {
       global $SESS_DBH, $SESS_LIFE;

       $expiry = time() + $SESS_LIFE;
       $value = addslashes($val);
```

```
	$qry = "INSERT INTO sessions VALUES ('$key', $expiry, '$value')";
	$qid = mysql_query($qry, $SESS_DBH);

	if (! $qid) {
		$qry = "UPDATE sessions SET expiry = $expiry, value = '$value' WHERE
sesskey = '$key' AND expiry > " . time();
		$qid = mysql_query($qry, $SESS_DBH);
	}

	return $qid;
}

function sess_destroy($key) {
	global $SESS_DBH;

	$qry = "DELETE FROM sessions WHERE sesskey = '$key'";
	$qid = mysql_query($qry, $SESS_DBH);

	return $qid;
}

function sess_gc($maxlifetime) {
	global $SESS_DBH;

	$qry = "DELETE FROM sessions WHERE expiry < " . time();
	$qid = mysql_query($qry, $SESS_DBH);

	return mysql_affected_rows($SESS_DBH);
}

session_set_save_handler(
	"sess_open",
	"sess_close",
	"sess_read",
	"sess_write",
	"sess_destroy",
	"sess_gc");
?>
```

Y el script de testing es:

```
<?
/* ------------------------------------------------------------------------
 * test.php
 * ------------------------------------------------------------------------
 * PHP4 Customer Session Handler Test Script
 * Version 1.00
 * by Ying Zhang (ying@zippydesign.com)
 * Last Modified: May 21 2000
 */

/* default to DBM handler */
if (! isset($handler)) {
	$handler = "dbm";
}

/* default action is increment */
if (! isset($action)) {
	$action = "increment";
}

/* load up the appropriate session handling script, depending on the handler */
```

```
if ($handler == "dbm") {
	include("session_dbm.php");

} elseif ($handler == "mysql") {
	include("session_mysql.php");

} else {
	echo "<li>Unrecognized handler ($handler)";
	die;
}

/* start the session and register a simple counter */
session_start();
session_register("count");

/* figure out what we should do, depending on the action */
switch ($action) {
	case "increment" :
		$count = isset($count) ? $count + 1 : 0;
		break;

	case "destroy" :
		session_destroy();
		break;

	case "gc" :
		$maxlife = get_cfg_var("session.gc_maxlifetime");
		sess_gc($maxlife);
		break;

	default:
		echo "<li>Unknown action ($action)";
		break;
}
?>

<h1>Session Test Script</h1>
<ul>
<li>Handler: <b><?=$handler?></b>
<li>Action: <b><?=$action?></b>
<li>Count: <b><?=$count?></b>
</ul>

<hr size=1>
<form>
<table>
<tr>
	<td>Handler:</td>
	<td>
		<select name="handler">
		<option value="dbm">DBM</option>
		<option value="mysql">MySQL</option>
		</select>
	</td>
</tr>
<tr>
	<td>Action:</td>
	<td>
		<select name="action">
		<option value="increment">Increment</option>
		<option value="destroy">Session Destroy</option>
		<option value="gc">Force Garbage Collection</option>
		</select>
```

```
	</td>
</tr>
<tr>
	<td></td>
	<td><br><input type="submit"></td>
</tr>
</table>
</form>
```

Capítulo 8: Manejo de vectores.

En el capítulo introductorio al lenguaje se estudiaron los vectores como tipos de datos en PHP, se vio como crear y utilizar vectores y vectores asociativos, en este capítulo se estudian funciones nativas de PHP que facilitan la manipulación de vectores y matrices.

Colas y Pilas usando vectores en PHP.

PHP cuenta con instrucciones nativas para manipular vectores que permiten utilizar los mismos como pilas o colas, las instrucciones son:

variable=array_pop(array);

Elimina del vector el último elemento y lo devuelve en una variable.

array_push(array, variable);

Agrega un elemento al final del vector con el valor de la variable que se le pasa como segundo parámetro.

variable=array_shift(array);

Elimina del vector el primer elemento y lo devuelve en la variable.

array_unshift(array,variable);

Agrega el elemento pasado al principio del vector desplazando todos los valores.

Usando array_shift y array_pop se pueden implementar colas, usando array_push y array_pop tenemos una pila mientras que usando las 4 instrucciones podemos manejar facilmente una cola doble.

Funciones de sort

PHP dispone de varias modalidades de sort para vectores, las funciones son:

sort(array);

Ordena un vector según los valores de sus elementos, si es un vector asociativo considera claves y valores como elementos comunes (no los distingue). Ordena en orden ascendiente.

rsort(array);

Idem anterior pero ordena en orden descendiente.

asort(array);

Ordena un vector según los valores de sus elementos pero manteniendo las asociaciones clave-valor. Ordena los pares ordenados clave-valor según “valor”

arsort(array);

Idem anterior pero en orden descendiente.

ksort(array);

Ordena un vector asociativo por los valores de sus “claves” teniendo en cuenta las asociaciones clave-valor.

krsort(array);

Idem anterior pero en orden descendiente.

usort(array,funcion);

Dado un array y una función de comparación ordena los “valores” del array usando la funcion provista para comparar elementos. La función debe recibir dos elementos y devolver 1 si el primero es mayor, -1 si el segundo es mayor o bien 0 si los elementos son iguales. Ejemplo:

```
function cmp ($a, $b) {
   if ($a == $b) return 0;
   return ($a > $b) ? -1 : 1;
}
$a = array (3, 2, 5, 6, 1);
usort ($a, cmp);
```

uksort(array,funcion);

Ordena un vector asociativo por “clave” usando para comparar las claves la función pasada como parámetro.

uasort(array,funcion);

Ordena un vector por los “valores” de sus elementos preservando la relación clave-valor de un array asociativo usando para ordenar la funcion provista por el usuario.

Ejemplos:

$vector=array(“d”=>”banana”, “a”=>”limon”,”c”=>”pera”, “b”=>”anana”);

Función	Resultado
sort($vector)	“a”,“anana”,”b”,”banana”,”c”,”d”,”limon”,”pera”
rsort($vector)	“pera”,”limon”,”d”,”c”,”banana”,”b”,”anana”,”a”
asort($vector)	“b”,”anana”,”d”,”banana”,”a”,”limon”,”c”,”pera”
arsort($vector)	“c”,”pera”,”a”,”limon”,”d”,”banana”,”b”,”anana”
ksort($vector)	“a”,”limon”,”b”,”anana”,”c”,”pera”,”d”,”banana”
krsort($vector)	“d”,”banana”,”c”,”pera”,”b”,”anana”,”a”,”limon”

Funciones para manipular vectores:

Padding:

array=array_pad(array,pad_size,pad_value);

Completa el array pasado con pad_value hasta que el vector tenga pad_size elementos, si pad_size es positivo completa agregando elementos hacia la derecha, si es negativo completa hacia la izquierda. El vector no es modificado, devuelve el vector resultado.

```
$input = array (12, 10, 9);
$result = array_pad ($input, 5, 0);  // result is array (12, 10, 9, 0, 0)
$result = array_pad ($input, -7, -1); // result is array (-1, -1, -1, -1, 12, 10, 9)
$result = array_pad ($input, 2, "noop");  // not padded
```

List:

List en realidad no es una instrucción sino una construcción especial del lenguaje que permite asignar a un grupo de variables los elementos de un vector.

Ejemplo:

```
$vector=array(1,2);
list($a,$b)=$vector;   //$a=1, $b=2
```

Si el vector tiene más elementos que las variables que se usan en list entonces el último elemento de list será un vector con todos los elementos que quedaban en el array (la asignación se hace de izquierda a derecha).

Merge:

```
array=array_merge(array1,array2,....);
```

Si los vectores son asociativos hace un merge de los vectores en donde si 2 o más vectores tienen la misma clave sólo una queda en el vector resultado. Si los vectores no son asociativos (indexados por número) entonces el resultado tiene todos los elementos de los "n" vectores pasados concatenados.

Sub-Vectores:

```
array=array_slice(array,offset[,cantidad]);
```

Devuelve un sub-vector del vector pasado a partir del offset indicado y con la cantidad de elementos indicada, si cantidad no se especifica devuelve todos los elementos desde offset hasta el fin del vector.

```
$vec=array(10,6,7,8,23);

$res=array_slice($vec,1,3);   //deja en la variable $res 6,7,8
```

Count:

```
$cantidad=count($vector);
```

Devuelve la cantidad de elementos de un vector.

Splice:

```
array=array_splice(array,offset,[cantidad,array_reemplazo]);
```

Si array_reemplazo no se pasa como parámetro entonces elimina del vector pasado la cantidad de elementos indicada a partir del offset indicado, si array_reemplazo existe dichos elementos son reemplazados por aquellos del vector array_reemplazo.
Si no se pasa cantidad se eliminan o reemplazan todos los elementos desde el offset indicado hasta el fin del vector.

Shuffle:

```
shuffle(array);
```

Desordena en forma aleatoria los elementos de un vector.

Pertenencia:

Boolean = in_array(variable,array);

Devuelve true/false según el elemento pasado pertenezca o no al vector.

Range:

array=range(low,high);

Crea un vector con los números correspondientes desde low hasta high.

Ejemplo:

$vec=range(6,12); //$vec=(6,7,8,9,10,11,12);

Reverse:

array=array_reverse(array);

Devuelve el vector revertido.

Compact:

array=compact(nombre_var1,nombre_var2,....,nombre_varn);

Crea un vector asociativo con las variables y sus valores donde las claves son los nombres de las variables y los valores el contenido de las mismas.

Ejemplo:
$ciudad=”miami”;
$edad=”23”;

$vec=compact(“ciudad”,”edad”);

Es equivalente a:

$vec=array(“ciudad”=>”miami”,”edad”,”23”);

Funcion Alfa:

PHP soporta una función muy usada en programación funcional que es conocida como función alfa, en php se denomina array_walk y permite aplicar una función a todos y cada uno de los elementos de un vector. La sintaxis es:

array_walk(array,funcion,variable_extra);

variable_extra es opcional, se aplica la función pasada como parámetro a cada uno de los elementos del vector, la función recibirá como parámetro en primer lugar el “valor” del elemento del array y en segundo lugar la “clave”, si el vector no es asociativo la clave es el numero de índice (0,1,2...). Si se pasa variable_extra que puede ser cualquier tipo de PHP incluyendo un objeto la función recibe dicha variable como tercer parámetro.

Funciones especiales para vectores asociativos:

array=array_keys(array)

Devuelve un vector con todas las claves de un vector asociativo.

array=array_values(array)

Devuelve un vector con todos los valores de un vector asociativo.

Funciones para recorrer vectores:

En PHP cada vector tiene asociado un puntero interno que apunta a un elemento del vector y que puede ser usado para recorrer vectores y otras operaciones, las funciones que operan con el puntero interno son:

reset(array);

Resetea el puntero interno al principio del array.

end(array);

Setea el puntero interno apuntando al último elemento del array

next(array);

Mueve el puntero al proximo elemento del array

prev(array);

Mueve el puntero al último elemento del array

current(array);

Devuelve el elemento apuntado actualmente por el puntero interno del array.

key(array);

Devuelve la clave (índice) del elemento apuntado actualmente por el puntero interno del array, si es un vector asociativo devuelve la clave del elemento actual. Si es un vector común devuelve el numero de índice del elemento actual.

array=each(array)

Devuelve un vector clave-valor con los valores correspondientes al elemento actual del array y además mueve el puntero al elemento siguiente, si es un vector asociativo devuelve clave-valor, si es un vector común devuelve indice-valor.

Each suele usarse en conjunto con list para recorrer un vector:

Ejemplo:

```
while(list($clave,$valor)=each($vector)) {
  //Hacer algo
}
```

Capitulo 9: Manejo de Strings y expresiones regulares.

A continuación se describe un resumen de las funciones mas importantes de PHP para manejo de strings.

Mayúsculas y minúsculas:

string=strtoupper(string);

Pasa un string a mayúsculas.

string=strtolower(string);

Pasa un string a minúsculas.

string=ucfirst(string);

Pasa a mayúscula el primer carácter de un string

string=ucwords(string);

Pasa a mayúsculas el primer carácter de cada palabra de un string (separadas por blancos, tabulaciones y saltos de línea)

Trimming:

string=chop(string);

Elimina blancos y saltos de línea a la derecha de un string dado.

string=ltrim(string);

Elimina blancos y saltos de línea a la izquierda de un string.

string=trim(string);

Elimina blancos y saltos de línea a derecha e izquierda de un string.

Comparaciones:

int=strpos(string1,string2);

Devuelve la posición de la primera ocurrencia de string 2 dentro de string1.

int=strspn(string1,string2);

Devuelve la longitud en caracteres de s1 contando desde el principio hasta que aparece un caracter en s1 que no esta en s2.

int=strcmp(string1,string2);

Compara dos strings y devuelve 1, 0 o –1 según sea mayor el primero, iguales o el segundo.

int=strcasecmp(string1,string2);

Idem anterior pero case-insesitive (no distingue mayúsculas y minúsculas)

int=strcspn(string1,string2);

Devuelve la longitud de s1 desde el principio hasta que aparece un caracter que pertenece a s2.

int=strstr(string1,string2);

Devuelve todos los caracteres de s1 desde la primera ocurrencia de s2 hasta el final.

int=stristr(string1,string2);

Idem anterior pero case-insensitive (no distingue mayúsculas de minúsculas)

int=similar_text(string1,string2,referencia);

Analiza la semejanza entre dos strings, devuelve la cantidad de caracteres iguales en los dos strings, si se pasa como tercer parámetro una referencia a una variable devuelve en la misma el porcentaje de similitud entre ambos strings de acuerdo al algoritmo de Oliver (1993).

Ejemplo:

similar_text($st1,$st2,&$porcentaje);

Funciones de Parsing:

array=explode(separator,string);

Devuelve un vector donde cada elemento del vector es un substring del string pasado particionado de acuerdo a un cierto caracter separador.

Ejemplo:

$st=”hola,mundo,como,estan”
$vec=explode(“,”,$st); //$vec=(“hola”,”mundo”,”como”,”estan”);

string=implode(separator,array);

Genera un string concatenando todos los elementos del vector pasado e intercalando separator entre ellos.

string=chunk_split(string,n,end);

end es opcional y por default es “\r\n”, devuelve un string en donde cada “n” caracteres del string original se intercala el separador “end”.

Ejemplo:

$st=”hola mundo”;
$st2=chunk_split($st,2,”,”);
//$st2=”ho,la, m,un,do”;

array=count_chars(string);

Devuelve un vector de 256 posiciones donde cada posición del vector indica la cantidad de veces que el caracter de dicho orden aparece en el vector.

string=nl2br(string);

Devuelve un string en donde todos los saltos de línea se han reemplazado por el tag
 de html.

string=strip_tags(string,string_tags_validos);

Devuelve un string eliminando del string original todos los tags html, si se pasa el segundo parámetro opcional es posible especificar que tags no deben eliminarse (solo hace falta pasar los tags de apertura)

ejemplo:

$st2=strip_tags($st1,"
 <table>");

Elimina todos los tags html de $st1 excepto
 , <table> y </table>

string=metaphone(string);

Devuelve una representación metafónica (similar a soundex) del string de acuerdo a las reglas de pronunciación del idioma ingles.

string=strtok(separador,string);

Dado un separador obtiene el primer "token" de un string, sucesivas llamadas a strtok pasando solamente el separador devuelven los tokens en forma sucesiva o bien falso cuando ya no hay mas tokens.

Ejemplo:

```
$tok=strtok($st,"/");
while($tok) {
  //Hacer algo
  $tok=strtok("/");
}
```

parse_string(string);

Dado un string de la forma "nombre=valor&nombre2=valor2&nombre3=valor3", setea las variables correspondientes con los valores indicados, ejemplo:

```
parse_string("v1=hola&v2=mundo");
//Seteta $v1="hola" y $v2="mundo"
```

Codificación y decodificación ASCII.

char=chr(int);

Devuelve el caracter dado su número ascii.

int=ord(char);

Dado un caracter devuelve su código Ascii.

Substrings:

string=substr(string,offset,longitud);

Devuelve el substring correspondiente al string pasado desde la posición indicada por offset y de la longitud indicada como tercer parámetro, si no se pasa el tercer parámetro se toman todos los caracteres hasta el final del string.

string=substr_replace(string,string_reemplazo,offset, longitud);

Idem anterior pero el substring seleccionado es reemplazado por string_reemplazo, si string_reemplazo es “” entonces sirve para eliminar una porción de un string.

Búsquedas y Reemplazos.

str_replace(string1,string2,string3);

Reemplaza todas las ocurrencias de string1 en string3 por string2. Esta función no admite expresiones regulares como parámetros.

string=strtr(string1,string_from,string_to);

Reemplaza en string1 los caracteres en string_from por su equivalente en string_to (se supone que string_from y string_to son de la misma longitud, si no lo son los caracteres que sobran en el string mas largo se ignoran)

Ejemplo:

```
$st=”hola mundo”
strtr($st,”aeiou”,”12345”);
//$st=”h4la m5nd4”
```

array=split(pattern,string);

Idem a explode pero el separador puede ser ahora una expresión regular.

boolean=ereg(pattern,string,regs);

Devuelve true o false según si el string matchea o no una expresión regular dada, el tercer parámetro es opcional y debe ser el nombre de un vector en donde se devolverán los matches de cada paréntesis de la expresion regular si es que la misma tiene paréntesis.

boolean=eregi(pattern,string,regs);

Idem anterior pero case-insensitive.

ereg_replace(pattern_from,string_to,string);

Reemplaza todas las ocurrencias de una expresión regular en string por el contenido de string_to.

eregi_replace(pattern_from,string_to,string);

Idem anterior pero no considera mayúsculas y minúsculas para la búsqueda de la expresión regular en el string.

Sintaxis básica de una expresión regular:

Los símbolos especiales "^" y "$" se usan para matchear el principio y el final de un string respectivamente. Por ejemplo:

"^el"	Matchea strings que empiezan con "el"
"colorin colorado$"	Matchea strings que terminan en "colorin colorado"
"^abc$"	String que empieza y termina en abc, es decir solo "abc" matchea
"abc"	Un string que contiene "abc" por ejemplo "abc" ,"gfabc", "algoabcfgeh", etc...

Los símbolos "*" , "+" y "?" denotan la cantidad de veces que un caracter o una secuencia de caracteres puede ocurrir. Y denotan 0 o más, una o más y cero o una ocurrencias respectivamente.
Por ejemplo:

"ab*"	Matchea strings que contienen una "a" seguida de cero o mas "b" Ej: "a", "ab", "cabbbb", etc
"ab+"	Matchea strings que contienen una "a" seguida de una o mas "b"
"ab?"	Matchea strings que contienen una "a" seguida o no de una "b" pero no mas de 1.
"a?b+$"	Matchea "a" seguida de una o mas "b" terminando el string.

Para indicar rangos de ocurrencias distintas pueden especificarse la cantidad máxima y mínima de ocurrencias usando llaves de la forma {min,max}

"ab{2}"	Una "a" seguida de exactamente 2 "b"
"ab{2,}"	Una "a" seguida de 2 o mas "b"
"ab{3,5}"	Una "a" seguida de 3 a 5 "b" ("abbb", "abbbb", "abbbbb")

Es obligatorio especificar el primer número del rango pero no el segundo. De esta forma
+ equivale a {1,}. * equivale a {0,} y ? equivale a {0,1}

Para cuantificar una secuencia de caracteres basta con ponerla entre paréntesis.

"a(bc)*"	Matchea una "a" seguida de cero o mas ocurrencias de "bc" ej: "abcbcbc"

El símbolo "|" funciona como operador "or"

"hola|Hola"	Matchea strings que contienen "hola" u "Hola"
"(b|cd)ef"	Strings que contienen "bef" o "cdef"
"(a|b)*c"	Secuencias de "a" o "b" y que termina en "c"

El carácter "." matchea a cualquier otro caracter.

"a.[0-9]"	Matchea "a" seguido de cualquier caracter y un dígito.
"^.{3}$"	Cualquier string de exactamente 3 caracteres.

Los corchetes se usan para indicar que caracteres son validos en una posición única del string.

"[ab]"	Matchea strings que contienen "a" o "b"
"[a-d]"	Matchea strings que contienen "a", "b" , "c" o "d"
"^[a-zA-Z]"	Strings que comienzan con una letra.
"[0-9]%"	Un dígito seguido de un signo %

También puede usarse una lista de caracteres que no se desean agregando el símbolo "^" dentro de los corchetes, no confundir con "^" afuera de los corchetes que matchea el principio de línea.

"[^abg]"	Strings que NO contienen "a" , "b" o "g"

“[^0-9]” Strings que no contienen dígitos

Los caracteres “^.[$()|*+?{\” deben escaparse si forman parte de lo que se quiere buscar con una barra invertida adelante. Esto no es válido dentro de los corchetes donde todos los caracteres no tienen significado especial.

Ejemplos:

Validar una suma monetaria en formato: “10000.00”, “10,000.00” .,“10000” o “10,000” es decir con o sin centavos y con o sin una coma separando tres dígitos.

^[1-9][0-9]*$

Esto valida cualquier número que no empieza con cero, lo malo es que “0” no pasa el test. Entonces:

^(0|[1-9][0-9]*)$

Un cero o cualquier número que no empieza con cero. Aceptemos también un posible signo menos delante.

^(0|-?[1-9][0-9]*)$

O sea cero o cualquier número con un posible signo “-“ delante.

En realidad podemos admitir que un número empiece con cero para una cantidad monetaria y supongamos que el signo “-“ no tiene sentido, agreguemos la posibilidad de decimales:

^[0-9]+(\.[0-9]+)?$

Si viene un “.” debe esta seguido de un dígito, 10 es válido pero “10.” no
Especifiquemos uno o dos dígitos decimales:

^[0-9]+(\.[0-9]{1,2})?$

Ahora tenemos que agregar las comas para separar de a miles.

^[0-9]{1,3}(,[0-9]{3})*(\.[0-9]{1,2})?$

O sea un conjunto de 1 a 3 dígitos seguido de uno más conjuntos de “,” seguido de tres dígitos. Ahora hagamos que la coma sea opcional.

^([0-9]+|[0-9]{1,3}(,[0-9]{3})*)(\.[0-9]{1,2})?$

Y de esta forma podemos validar números con los 4 formatos validos en una sola expresión.

Capítulo 10: Generación dinámica de imágenes.

PHP provee la posibilidad de generar imágenes dinámicamente y de incluir estas imágenes en una página web, esto se hace utilizando funciones de una biblioteca denominada “GD” que viene compilada en forma default en php4, según la versión de GD la biblioteca permita generar imágenes GIF o PNG.
Una vez generada dinámicamente la imagen es posible transmitirla directamente al browser o guardarla en disco para luego levantarla usando un tag <IMG> de html.

Creación de una Imagen.

image_handler=ImageCreate($x,$y);

Crea una Imagen de tamaño X por Y pixels y devuelve un handler a la imagen en $IM (se maneja el handler a la imagen en el resto de las funciones que manipulan la imagen, como si fuera un archivo).

Una vez creada la imagen PHP provee funciones para dibujar rectángulos, arcos, texto y demás elementos en la imagen:

Otras variantes para crear una imagen consisten en crear la imagen a partir de una imagen existente en el disco de forma tal de poder modificarla:

int=imagecreatefromgif(path);
int=imagecreatefrompng(path);
int=imagecreatefromjpg(path);

Al igual que ImageCreate estas funciones devuelven un ImageHandler.

Creación y alocación de colores.

Para utilizar colores en una imagen es necesario en primer lugar crear el color y alocarlo en la imagen, esto se hace con ImageColorAllocate de la siguiente forma:

color_handler=ImageColorAllocate(image_handler,int_rojo,int_verde,int_azul);

La función recibe una image_handler en donde alocar el color y los valores decimales de la cantidad de rojo, verde y azul del color (0 a 255), devuelve un color_handler que puede ser usado en cualquiera de las funciones que veremos a continuación y utilizan colores.

Funciones para creación de objetos en la imagen.

ImageRectangle(image_handler, x1,y1,x2,y2,color_handler);

Dibuja un rectángulo desde la coordenada x1,y1 (0,0 es la esquina superior izquierda de la imagen) hasta la coordenada x2,y2 del color indicado por color_handler (previamente alocado con ImageColorAllocate).

ImageFilledRectangle(image_handler, x1,y1,x2,y2,color_handler);

Idem anterior pero dibuja el rectángulo relleno con el color indicado.

int imagearc (image_handler, cx, cy, ancho, alto, angulo_comienzo, angulo_fin, color_handler)

Dibuja un arco de elipse centrado en cx,cy con el ancho y alto especificado (sin son iguales la elipse es una circunferencia) y desde el ángulo de comienzo al ángulo de fin (en grados 0 a 360). El arco se dibuja con el color indicado.

ImageDashedLine(image_handler, x1,y1,x2,y2,color_handler);

Dibuja una línea puntuada entre las coordenadas especificadas y con el color indicado.

ImageFill(image_handler,x,y,color_handler);

Pinta con el color indicado a partir de la coordenada x,y y con el color indicado, llena con el color indicado.

ImagePolygon(image_handler,array_puntos,cantidad_puntos,color_handler);

Dibuja un polígono usando un vector de puntos de la forma (x0,y0,x1,y1,x2,y2,....etc) el parámetro cantidad_puntos indica cuantos puntos considerar para crear el polígono.

ImageFilledPolygon(image_handler,array_puntos,cantidad_puntos,color_handler);

Idem anterior pero el polígono además se rellena con el color indicado.

ImageLine(image_handler, x1,y1,x2,y2,color_handler);

Dibuja una línea desde x1,y2 hasta x2,y2 con el color indicado.

Manejo de Colores:

color_handler=ImageColorAt(iamge_handler,x,y);

Devuelve el color handler correspondiente al color del pixel especificado.

color_handler=imagecolorclosest (image_handler, int_rojo,int_verde, int_azul)

Devuelve el color alocado más cercano al color indicado en RGB por las cantidades de rojo, verde y azul (en decimal)

color_handler=imagecolorexact (image_handler, int_rojo,int_verde, int_azul)

Idem anterior pero devuelve el color_handler del color pasado si el color no esta alocado en la imagen devuelve –1.

color_handler=imagecolorresolve (image_handler, int_rojo,int_verde, int_azul)

Es una mezcla de las dos anteriores, esta función siempre devuelve un color_handler, o bien el color exacto alocado en la imagen o bien el color más cercano.

int=ImageColorsTotal(image_handler)

Devuelve la cantidad total de colores de la imagen.

imagecolortransparent(image_handler,color_handler)

Setea el color indicado por el handler como transparente para la imagen.

ImageCopy (image_handler_dest,image_handler_origen,x_dest, y_dest, origen_x,origen_y, ancho,alto)

Copia una porción de imagen desde la coordenada origen_x, origen_y con el ancho y alto especificado desde image_handler_origen hacia image_handler_dest en la coordenada x_dest, y_dest.

imagecopyresized (image_handler_dest, image_handler_origen,dest_x,dest_y,origen_x,origen_y,dest_ancho, dest_alto, origen_ancho, origen_alto)

Copia con opción de achicar o agrandar una porción de la imagen hay que especificar la imagen origen, la imagen destino, la coordenada desde donde copiar en la imagen origen, la coordenada a donde copiar en la imagen destino, y el ancho y alto tanto en el origen como en el destino.

Manejo de texto.

imagestring(image_handler,font_number, x, y, string, color_handler)

Coloca un string en la imagen, si font=0 se usa el font default, si font es 1,2,3,4, o 5 se usa un font predefinido.X e Y son las coordenadas donde dibujar el string y especifican la esquina superior izquierda del string.

imagestringup(image_handler,font_number, x, y, string, color_handler)

Idem anterior pero el string se dibuja en forma vertical.

array imagettftext (image_handler,size, angulo,x, y, color_handler, font_path, string)

Dibuja un string en la imagen usando un font treu-type, el font en formato nombre.ttf debe guardarse en algún lugar del file-system que se especifica con font_path (ej.: /fonts/arial.ttf). X e Y son las coordenadas de la esquina inferior izquierda del string. Angulo es el ángulo con el cual se dibuja el string (0=de izquierda a derecha en forma horizontal). Size indica el tamaño en puntos del texto a usar. Devuelve un vector de 8 elementos representando los 4 puntos que delimitan al string de la forma:
izquierda_arriba, derecha_arriba, abajo_izquierda, abajo_derecha. (cada esquina esta representada por dos coordenadas: x e y)

array imagettfbbox (size, ángulo, font_path, string)

Determina el tamaño que ocupara el string en la imagen y devuelve un vector de 8 elementos con el mismo formato descripto en la función anterior.

Generación de la Imagen.

ImageInterlace(image_handler,boolean)

Determina si la imagen será interlazada o no 1=interlazada, 0=no interlazada.

imagegif (image_handler, path)

Dado un image_handler genera un archivo gif con la imagen correspondiente, el archivo puede luego incluirse con un tag <IMG> para mostrarse en una página.

Si no es necesario almacenar la imagen es posible llamar a imagegif de la forma:

imagegif (image_handler)

En cuyo caso la imagen generada es transmitida directamente al browser, para ello antes es necesario enviar al browser un header indicando que se va a recibir un gif.

Ejemplo:

header("Content-Type: image/gif");
ImageGIF($IM);

Ejemplo 1:

```
<?php
Header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = ImageColorAllocate ($im, 0, 0, 0);
$white = ImageColorAllocate ($im, 255, 255, 255);
ImageTTFText ($im, 20, 0, 10, 20, $white, "/path/arial.ttf","Testing...
Omega: &#937;");
ImageGif ($im);
ImageDestroy ($im);
?>
```

Este ejemplo genera un gif con un texto usando un font true-type y lo transmite directamente al browser. Es importante que antes y después de los tags <? y ?> no existan espacios en blanco o saltos de línea ya que en ese caso PHP transmitirá al browser estos caracteres y la imagen recibida se vera rota, o sea que debe asegurarse que lo único que recibe el browser es el código de la imagen.

Ejemplo 2:

El siguiente script recibe como parámetros un tamaño x (x), un tamaño y (y), un porcentaje (per), un color de fondo (bg), un color de frente (fg) y dibuja una barra de porcentaje con el porcentaje indicado. Los colores se pasan en hexadecimal. Si no se indica algún parámetro se toman valores default.

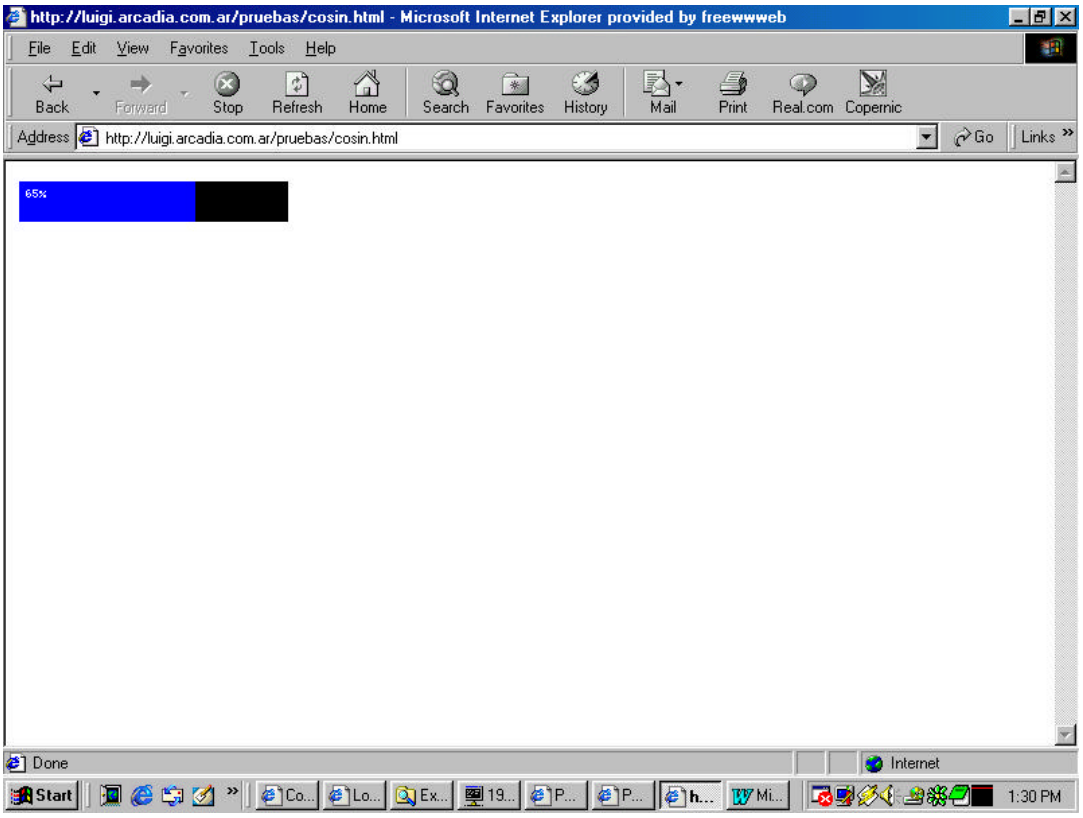
```
<?
//Parámetros:
//Tamaño x, tamaño y, porcentaje de la barra,
//colores ( en notación #FFFFFF)
//En las variables: $x,$y,$per,$bg,$fg

if(!isset($x)){$x=140;}
if(!isset($y)){$y=20;}
if(!isset($per)){$per=75;}
if(!isset($bg)){$bg="#FF0000";}
if(!isset($fg)){$fg="#0000FF";}

$rb=base_convert(substr($bg,1,2),16,10);
$gb=base_convert(substr($bg,3,2),16,10);
$bb=base_convert(substr($bg,5,2),16,10);
$rf=base_convert(substr($fg,1,2),16,10);
$gf=base_convert(substr($fg,3,2),16,10);
$bf=base_convert(substr($fg,5,2),16,10);
$xp=round(($per/100)*$x);

$IM=ImageCreate($x,$y);
$colbg=ImageColorAllocate($IM,$rb,$gb,$bb);
$colfg=ImageColorAllocate($IM,$rf,$gf,$bf);
//$f=ImageLoadFont("fonts/ARIAL.TTF");
$blanco=ImageColorAllocate($IM,255,255,255);
ImageFilledRectangle($IM,0,0,$x,$y,$colbg);
ImageFilledRectangle($IM,0,0,$xp,$y,$colfg);
ImageString($IM,0,5,5,"$per%",$blanco);
```

```
header("Content-Type: image/gif");
ImageGIF($IM);
?>
```

La forma de llamar al script es por ejemplo:

<IMG src="barra.php?x=200&y=30&per=65&bg=#FF4356&fg=#456890">

Y el resultado es:



Como puede verse la llamada es mediante un tag <IMG> con la salvedad de que la imagen que se llama no existe en el disco sino que es un script php que genera dinámicamente la imagen y la entrega al browser.

Utilizando estas funciones de PHP es posible crear gráficos estadísticos en función de datos de la base, modificar imágenes agregando texto, sobreimpresos y realizar varios efectos en función de datos ingresados por el usuario como por ejemplo mapas dinámicos, gráficos de torta, etc...

Capítulo 11: Manejo de fechas

PHP provee varias funciones para manipulación, validación y formateo de fechas, el formato interno para representar una fecha en PHP es el usado por Unix o sea una cantidad de segundos a partir de una fecha definida como EPOCH. Las funciones más importantes son las que describimos a continuación:

Date:

string=date(string_formato,time);

El segundo parámetro es opcional, si se pasa debe ser una fecha en formato de representación interna, si no se pasa se toma como fecha la fecha actual. El formato es un string de formato libre en el cual ciertos caracteres tienen un significado especial y son reemplazados por ciertos valores:

- a - "am" o "pm"
- A - "AM" o "PM"
- d – día del mes en dos dígitos con cero adelante si es necesario
- D – día de la semana en ingles en formato de tres letras Ej: “fri”
- F – Nombre del mes en ingles Ej: “January”
- h – Hora en formato de 12 horas: 01 a 12
- H – Hora en formato de 24 horas: 00 a 23
- g – Hora en formato de 12 horas sin ceros adelante: 1 a 12
- G – Hora en formato de 24 horas sinc ceros adelante: 0 a 23
- i – Minutos en dos dígitos: 00 a 59
- j – día del mes sin ceros adelante: 1 a 12
- l - día de la semana en ingles completo Ej: “Friday”
- L – Boolean que indica si el año es bisiesto (true=es, false=no)
- m – Número de mes 01 a 12
- n – Número de mes sin ceros adelante 1 a 12
- M – Nombre del mes en ingles en tres letras Ej: “Jan”
- s – Segundos 00 a 59
- S – Sufijo ordinal en ingles para el número de día (th,nd,st)
- t – Número de días para el mes actual 1 a 31
- **U – Segundos pasados desde EPOCH (formato de representación interno)**
- w – día de la semana en formato numérico (0=domingo)
- Y – Año en 4 dígitos
- y – Año en 2 dígitos
- z – Día del año 1 a 365

Ejemplo:

```
date(“Hoy es d/m/Y y la hora es: H:i:s”)
```

Queda algo de la forma “Hoy es 10/05/2000 y la hora es 15:06:29”

Para obtener la representación interna de una fecha dado el día, mes, año, horas, minutos y segundos se usa la función mktime:

int=mktime (hora, minutos, segundos, mes, día, año)

Devuelve la cantidad de segundos pasados desde el epoch, luego puede usarse este valor devuelto como segundo parámetro de “Date” para formatear la fecha en el formato que se desee.

Otras funciones:

boolean=checkdate(mes, día, año)

string=microtime()

Devuelve una representación de la hora actual incluyendo microsegundos, el string que devuelve tiene el formato “microsegundos segundos” y luego puede hacerse un explode del mismo tomando el espacio como separador para obtener los microsegundos.

La siguiente clase implementa timers con precisión de microsegundos que pueden usarse para medir duraciones de tiempo con gran precisión, lo cual es útil por ejemplo para realizar un “profile” de un script en php4 midiendo la duración de distintas partes del mismo (consultas a la base de datos, etc...)

```
// start('name') inicializa el timer con o sin nombre.
// stop('name')  para el timer
// current('name') para el timer y devuelve el tiempo transcurrido
//
class Timer {
  var $ss_timing_start_times;
  var $ss_timing_stop_times;


  function start($name='default'){
    $this->ss_timing_start_times[$name]=explode(' ',microtime());
  }

  function stop($name='default'){
    $this->ss_timing_stop_times[$name]=explode(' ',microtime());
  }

  function current($name='default') {
    if(!isset($this->ss_timing_start_times[$name])){
      return 0;
    }
    if(!isset($this->ss_timing_stop_times[$name])){
    $stop_time=explode(' ',microtime());
    } else {
    $stop_time=$this->ss_timing_stop_times[$name];
    }
    $current=$stop_time[1]-$this->ss_timing_start_times[$name][1];
    $current+=$stop_time[0]-$this->ss_timing_start_times[$name][0];
    return $current;
  }
}
?>
```

Ejemplo de uso:

```
$tim=new Timer();
$tim->start();
//codigo.
$tiempo=$tim->current();
```

Si se quieren usar múltiples timers simultáneos puede pasarse un “nombre” de timer a las funciones start, stop y current para diferenciar distintos timers y usarlos en forma independiente.

Capítulo 12: Archivos DBM

El formato DBM de archivos permite implementar una pseudo base de datos usando archivos planos, el formato que es standard tiene la ventaja de que los archivos DBM creados pueden ser fácilmente compartidos por otras aplicaciones que soporten el formato DBM (por ejemplo "C", "Perl","Python" y otros lenguajes tienen bibliotecas para manipular DBMs).
Básicamente una archivo DBM es un archivo común en el cual mediante un formato interno es posible almacenar pares de tipo "clave"-"valor". En palabras mas simples cada archivo DBM puede verse como una tabla de solo 2 columnas de una base de datos.

Las funciones de PHP para manejo de archivos DBM son:

dbm_handler=dbmopen(path,modo);

Abre un archivo dbm cuyo path recibe como parámetro, devuelve un handler al archivo (a ser usado por las demás funciones de dbm). Modo puede ser:

- "r" – Lectura solamente
- "w" – Lectura – escritura
- "n" – Lectura escritura, si existe el archivo lo trunca, si no existe lo crea
- "c" – Lectura escritura, si existe lo usa, si no existe lo crea

boolean=dbmexists(dbm_handler, clave);

Devuelve true/false según una clave exista o no en un archivo dbm.

boolean=dbminsert(dbm_handler,clave,valor);

Devuelve 0 si el insert fue exitoso (false!), 1 si la clave ya existía y –1 si no se puede escribir en el archivo.

string=dbmfetch(dbm_handler, clave);

Devuelve el valor asociado con la clave pasada, es conveniente antes usar dbmexists para chequear que la clave exista en el archivo dbm.

dbmreplace(dbm_handler, clave, valor);

Si la clave ya existe reemplaza su valor en el archivo dbm por el valor pasado, si la clave no existe la crea con el valor pasado. (Funciona como un insert que no da error si la clave ya existe)

boolean=dbmdelete(dbm_handler, clave);

Elimina un registro "clave"-"valor" de un archivo dbm. Devuelve falso si la clave no existía en el archivo DBM.

string=dbmfirstkey(dbm_handler);

Devuelve la primera clave de un archivo dbm.

string=dbmnextkey(dbm_handler);

Devuelve la próxima clave de un archivo dbm, en conjunto con dbmfirstkey puede usarse para recorrer todos los registros de un archivo dbm. Devuelve falso cuando no quedan más registros en el archivo.

Capítulo 13: Funciones matemáticas.

Constantes predefinidas:

Tabla 1. Constantes Matemáticas

Constante	Valor	Descripción
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)

Funciones:

Función	Descripción
var=abs(var);	Valor absoluto.
var=acos(var);	Arco coseno
var=atan(var);	Arco tangente
var=asin(var);	Arco seno
string=base_convert(string,base_from,base_to);	Realiza el pasaje de base correspondiente
var=bindec(string_binario);	Pasa de base 2 a decimal
int=ceil(float);	Redondea hacia arriba un número decimal (función techo)
var=cos(var);	Coseno
string=decbin(var);	Pasa de base 10 a base 2
string=dechex(var);	Pasa de base 10 a hexadecimal.
string=decoct(var)l	Pasa de base 10 a octal
var=deg2rad(var);	Pasa de grados (degrees) a radianes
var=exp(var);	Devuelve e^var (función exponencial)
var=floor(var);	Redondea un número decimal hacia abajo (función piso)
var=hexdec(string);	Pasa de hexadecimal a base 10
var=log(var);	Logaritmo natural (base e)
var=log10(var);	Logaritmo en base 10
var=octdec(string);	Pasa de octal a base 10
var=max(array);	Devuelve el elemento máximo de un vector
var=max(var1,var2,...,varn);	Devuelve el elemento máximo
var=min(array);	Devuelve el elemento mínimo de un vector
var=min(var1,var2,.....,varn);	Devuelve el elemento mínimo
var=pow(base,exponente);	Devuelve base^exponente
var=rad2deg(var);	Convierte radianes en grados (degrees)
var=round(var);	Redondea un número no entero a su valor entero más cercano
var=rand(min,max);	Genera un número random entre los valores pasados

srand(var);	Inicializa la semilla del algoritmo de generación de números al azar. En general: srand((double)microtime()*1000000);
var=sin(var);	Función seno
var=tan(var);	Función tangente
var=sqrt(var);	Devuelve la raíz cuadrada de un número

Capítulo 14: Ejecución de programas externos y scripting.

Es posible desde PHP invocar a un programa externo de forma tal de utilizar algún script externo para obtener resultados que luego sean utilizados en un script php.

string=escapeshellcmd(string)

Escapea todos los caracteres que puedan resultar peligrosos en un comando que va a pasarse al shell. En varias ocasiones un cierto input ingresado por el usuario es pasado a un programa externo para cumplir una determinada función. Supongamos que el usuario ingresa un nombre y password y un script php debe pasarle esos datos a un programa externo para cierta validación. El comando podría ser:

$comando="/usr/bin/validator $user $password";

Pero que pasa si el usuario ingresa como password: "pepe;rm –rf /*"; Entonces el comando quedaría:

"/usr/bin/validator nombre pepe;rm –rf/*""

Y al ejecutarse además de hacer lo que el script debe hacer el shell eliminara todos los archivos del disco (ugh!), la función escapeshellcmd evita esto anulando todos los casos peligrosos para la llamada a un comando.

Las funciones de ejecución de comandos son:

string=exec(command, array, var);

Los dos últimos parámetros son opcionales. Ejecuta un comando pasándoselo al shell y devuelve la última línea devuelta por el comando en su standard otuput, si se pasa un nombre de vector como segundo parámetro devuelve cada línea de salida del comando en un elemento del vector. Si se pasa una variable como tercer parámetro devuelve alli el resultado del comando al shell.

passthru(command,var);

La variable es opcional y recibe el valor de retorno del comando, passthru ejecuta el comando y redirecciona su salida al browser en forma directa. Esto es útil por ejemplo para programas externos que generan una imagen o algo similar (antes hay que enviar el header correspondiente).

Uso de php como lenguaje de scripting:

PHP puede usarse tanto como un modulo del webserver como también como lenguaje de scripting interpretado desde la línea de comandos (en cuyo caso funciona con cualquier web server que soporte el protocolo CGi aun cuando no soporte php como modulo). Para usar php desde la línea de comandos basta con compilar una versión de php en la cual no se le pasa la opción de compilarse como modulo de Apache, de esta forma se generara un binario "php" que es el interprete php.

Un script php se puede escribir entonces de la forma:

```
#!/usr/bin/php
<?
  print("Hola mundo\n");
?>
```

La primera línea es la que se conoce en Unix como sheebang line e indica el path del interprete para el código que sigue a continuación, en nuestro caso el binario php reside en /usr/bin/ pero podría estar en otro lado.

Luego escribimos un script standard, manteniendo el código php entre <? y ?>, todo lo que no este entre estos símbolos pasa directamente al standard output del script.

El script se invoca desde la línea de comandos como un programa ejecutable normal (dándole permiso de ejecución), también puede usarse la línea de comandos para probar un script que no tiene la línea sheebang, por ejemplo un script normal que usamos en un web site:

/usr/bin/php /path/nombre.php

Esto es muchas veces útil para chequear cual es la salida de un script que no esta funcionando bien en un web server la salida por la línea de comando al standard output es útil para análisis y debug ya que no pasa por la interpretación del browser.

Es posible pasarle a un script php parámetros por la línea de comandos, por ejemplo:

script.php hola mundo

Estos parámetros serán recibidos por el script en el vector $argv, el elemento [0] de $argv es el nombre mismo del script por lo que los dos parámetros en este caso estarán en $argv[1] y $argv[2].

En algunos casos queremos probar desde la línea de comandos un script que recibe parámetros desde un formulario html, es decir usando método GET o POST. Para ello no sirve el formato anterior ya que el script desconoce la existencia de $argv. Lo que hay que hacer es poner los parámetros en la variable de ambiente "QUERY_STRING" de la forma: "nombre=valor&nombre2=valor2...." y luego invocar al script. Usando bash como shell esto es de la forma:

[path] $QUERY_STRING="hola=mundo&nombre=juan"
[path] $ export QUERY_STRING
[path] $ /usr/bin/php /path/script.php

Automáticamente el interprete de php se encarga de parsear la variable de ambiente QUERY_STRING y convertirla en variables de php por lo que el resultado del script será el mismo que el que produciría si lo llamáramos desde un form html con esas variables seteadas.

Capítulo 15: Manejo de HTTP en PHP.

Uno de los temas más importantes en todo lenguaje de scripting usado para generación dinámica de sitios web y aplicaciones web es el manejo del protocolo HTTP, conexiones, métodos GET y POST, uploads, headers, cache y demás alternativas. Todas estas funciones están bien soportadas en php de forma tal de tener desde el lneguaje un control completo sobre la forma en que el server interactua con el browser.

Headers.

Una de las funciones más importantes de PHP es la función "header" que sirve para enviar al browser un determinado header HTTP, por default en cuanto un script PHP usa una función de salida o transmite algo al browser php envía el header "Content-Type: text/html" al browser. Por eso es importante destacar que la funcion header solo puede usarse ANTES de realizar cualquier tipo de salida al browser.

Ejemplos:

```
header("Location: http://lugar/pepe.php");
```

Este es un header http que sirve para redireccionar al browser a otra página, script o URL, es muy util para scripts php que procesan datos recibidos desde un formulario o similar y luego en funcion de los datos redireccionan al browser a una página que por ejemplo puede mostrar errores en el ingreso de datos, aceptar los datos o simplemente volver a la página que llamo al script.

Este tipo de redirección sólo puede usarse si no se emitió ninguna salida al browser, si ya se emitió una salida al browser y es necesario redireccionar la página debemos generar desde PHP código JavaScript que transmitimos al browser y el browser si es capaz de interpretar JavaScript podrá redireccionarse a la página pasada, por ejemplo:

```
<?
print("Cosas anteriores");
if($redireccionar) {
 ?>
  <script>
  location.href=<?print($URL);?>
  </script>
 ?<
}
?>
```

El browser recibe por ejemplo:

```
<script>
location..href=http://www.yahoo.com
</script>
```

Y automaticamente se redirecciona al URL pasado, el uso de JavaScript desde PHP es importante (ver el capítulo de programacion en el cliente desde el server)

Otro header importante que ya vimos es el de imagen que enviamos antes de pasarle al browser una imagen generada dinámicamente.

Una de las funciones importantes de los headers HTPP es controlar el comportamiento del cache del browser, en numerosas ocasiones tendremos que codificar scripts que generan una cierta salida que no es deseable que sea cacheable, si no impedimos que el browser cachee el resultado del script el usuario tendrá que refrescar manualmente la página desde el browser para ver el nuevo resultado. Los siguientes headers http impiden que el browser cachee una página y sirven tanto para Internet Explorer como para Netscape Navigator:

```
header("Cache-Control: no-cache, must-revalidate");
header("Pragma: no-cache");
header("Expires: Mon,26 Jul 1997 05:00:00 GMT");
```

El protocolo HTTP dispone de un mecanismo de autorización básico mediante el cual puede pedirse al browser que promptee al usuario por un usuario y password, luego estos datos se transmiten al web-server y el mismo debe encargarse de autorizar al usuario o no según corresponda. Desde PHP podemos generar los headers correspondientes para que el browser pida al usuario un usuario y password y luego validar estos datos desde el script php (usando una base de datos o lo que corresponda):

```
<?php
if(!isset($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\n";
exit;} else {
echo "Hello $PHP_AUTH_USER.<P>";
echo "You entered $PHP_AUTH_PW as your password.<P>";
}
?>
```

El header “WWW-Authenticate: Basic realm=\”My Realm\”” fuerza al browser a preguntar al usuario por un user y password, si el usuario cancela el script sigue su ejecución en la próxima línea (notar que enviamos un header de acceso no autorizado y termina el script”). Si el usuario ingresa un usuario y password el browser vuelve a invocar al script pasándole como variables $PHP_AUTH_USER y $PHP_AUTH_PW.

Control de la conexión usando PHP.

Internamente PHP mantiene un status de la conexión con el cliente durante la ejecución del script, el valor del status es habitualmente “NORMAL”, pero puede cambiar a “ABORTED” si el usuario presiona el botón de STOP o a “TIEMOUT” si es script sobrepasa el tiempo limite de ejecución que se fija en (/var/lib/php.ini). Es posible controlar que se hace si el usuario interrumpe la ejecución de un script con el botón de STOP usando dos funciones de PHP:

boolean=connection_aborted()

Devuelve true si la conexión fue abortada por el usuario presionando STOP, cerrando el browser o similar.

register_shutdown_function(nombre_funcion)

Esta funcion permite registrar el nombre de una función que se ejecutara en los siguientes casos:

a) Cuando el script intenta enviar algo al browser pero el usuario aborto la conexión.
b) Cuando termina normalmente la ejecución del script.

Para distinguir estos dos casos dentro de la función que se registre es importante usar la función connection_aborted para saber si la función se llama por la terminación normal del script o por conexión abortada. La función registrada con register_shutdown_function no puede generar ninguna salida (lo cual dificulta debuggearla), usualmente sirve para chequear consistencia en bases de datos, eliminar archivos temporales que pueden quedar abiertos, loggear un problema, registrar que la conexión fue abortada, hacer rollback de una transacción o similares.

Transacciones HTTP usando PHP.

Una modalidad muy en boga en estos días es el establecimiento de servidores de transacciones HTTP o HTTPS (usando SSL), muchas veces el propósito de estos servidores es hacer de gateway para solicitar servicios a una determinada red de servidores. Por ejemplo supongamos que el site www.super.com tiene 56 servidores, para poder interactuar con empresas externas www.super.com muchas veces tiene el problema de que un agente externo debe desencadenar una acción en uno de sus web servers, setear "n" servers para permitir que algunas direcciones en particular tengan acceso a cierta funcionalidad es muy complejo por lo que www.super.com decide instalar un servidor de transacciones, cualquiera tiene acceso a este servidor para solicitar servicios usando el protocolo HTTP y este servidor tiene acceso a los servers de www.super.com, el unico cambio en los 56 servers es permitir que el servidor de transacciones opere sobre ellos.

Para generar transacciones HTTP muchas veces es necesario que un scrip "simule" postear datos a un script como si los datos vinieran de un formulario HTML, a continuación sugerimos 2 distintas formas de generar transacciones POST desde un script sin necesidad de que un usuario submitee los datos:

Método 1: Usando CURL.

Curl es una pequeña utilidad que corre en Unix o Windows y permite generar paquetes HTTP de todo tipo, desde PHP podemos llamar a CURL como un programa externo para generar la transacción de la forma:

```
<?php
$data="hola=mundo";
$URL="luigi.arcadia.com.ar/pruebas/profo.php";
exec("/usr/local/bin/curl -m 120 -d \"$data\" http://$URL -L",$return_msg_array,$return_number);
for($i=0;$i<count($return_msg_array);$i++) {
  $results=$results.$return_msg_array[$i];
}
print("Resultado:$results\n");
```

Como vemos "curl" es llamado usando la función "exec" al ser un programa externo, $data tiene los datos a enviar en formato "nombre1=valor1&nombre2=valor2....", curl devuelve un vector donde cada elemento del vector es una línea de la salida que devuelve el server al recibir el request en modo POST.
En este script como vemos pegamos todas las líneas de salida y mostramos el resultado en pantalla.

Método 2: Usando la clase http_post.

Si no que quiere usar un programa externa podemos usar una clase que denominamos http_post y permite postear datos en un servidor, un ejemplo de uso de esta clase es:

```
<?
include("class_post.php");
$a=new http_post;
$a->set_action("http://luigi.arcadia.com.ar/pruebas/profo.php");
$a->set_element("hola","mundo");
$res=$a->send(0);
print("Resultado: $res");
?>
```

Como vemos en este script hacemos lo mismo que en el anterior, la diferencia principal radica en que esta clase devuelve como resultado "TODA" la salida del web server (incluyendo headers) mientras que CURL en la forma en que fue llamado no devuelve los headers.

La clase http_post es la siguiente:

```php
<?php
#
# http_post - PHP3 class for posting a 'form' from within a php3 script
# Version 0.5b
#
# Copyright 2000
# Alan van den Bosch (alan@sanguis.com.au)
# Sanguis Pty Ltd (acn 061 444 031)
#
# Licence:
# You are granted the right to use and/or redistribute this
# code only if this licence and the copyright notice are included
# and you accept that no warranty of any kind is made or implied
# by the author or Sanguis Pty Ltd.
#
#
# Methods:
#
# http_post()
# Constructor used when creating a new instance of the http_post class.
# Returns true on success.
# ie.
# $a=new http_post;
#
#
# set_server(string SERVER)
# Set the server of the URI you wish to post to. see also set_action()
# Returns true on success.
# ie.
# $a->set_server("127.0.0.1");
# or
# $a->set_server("www.somehost.org");
#
#
# set_port(string PORT)
# Set the tcp port of the URI you wish to post to. see also set_action()
# Returns true on success.
# ie.
# $a->set_port("8080");
#
#
# set_file(string FILENAME)
# Set the filename of the URI you wish to post to. see also set_action()
# Returns true on success.
# ie.
# $a->set_file("/incoming.php3");
#
#
# set_action(string ACTION)
# Set the URI you wish to post to.
# Returns true on success.
# ie.
# $a->set_action("http://www.somehost.org:8080/incoming.php3");
#
# set_enctype(string ENCTYPE)
# Set the encoding type used for the post. Can have the values
# "application/x-www-form-urlencoded" or "multipart/form-data"
```

```
# Returns true on success.
# ie.
# $a->set_enctype("multipart/form-data");
#
#
# set_element(string NAME, string VALUE)
# Set or update a single name/value pair to be posted
# Returns true on success.
# ie.
# $a->set_element("username","John Doe");
#
#
# set_element(array ELEMENTS)
# Set or update a number of name/value pairs to be posted
# Returns true on success.
# ie.
# $a->set_element(array("username" => "John Doe",
# "password" => "dead-ringer",
# "age" => "99"));
#
#
# set_timeout(integer TIMEOUT)
# Set the number of seconds to wait for the server to connect
# when posting. minimum value of 1 second.
# Returns true on success.
# ie.
# $a->set_timeout(10);
#
# show_post()
# Show the current internal state of an instance, for debugging.
# Returns true on success.
# ie.
# $a->show_post();
#
#
# send(boolean DISPLAY)
# Send the name/value pairs using the post method. The response
# can be echoed by setting DISPLAY to a true value.
# Returns a string containing the raw response on success, false
# on failure.
# ie.
# $a->send(1);
#


class http_post
{
function http_post(){
$this->_method= "post";
$this->_server=$GLOBALS[ "HTTP_HOST"];
$this->_file= "\\";
$this->_port= "80";
$this->_enctype= "application/x-www-form-urlencoded";
$this->_element=array();
$this->_timeout=20;
}
```

```
function set_server($newServer= ""){
if(strlen($newServer)<1)$newServer=$HTTP_HOST;
$this->_server=$newServer;
return 1;
}

function set_port($newPort= "80"){
$newPort=intval($newPort);
if($newPort < 0 || $newPort > 65535)$newPort=80;
$this->_port=$newPort;
return 1;
}

function set_file($newFile= "\\"){
$this->_file=$newFile;
return 1;
}

function set_action($newAction= ""){
$pat= "^((http://){1}([^:/]{0,}){1}(:([0-9]{1,})){0,1}){0,1}(.*)";

if(eregi($pat,$newAction,$sub)){
if(strlen($sub[3])>0)$this->_server=$sub[3];
if(strlen($sub[5])>0)$this->_port=$sub[5];
$this->_file=$sub[6];
return 1;
}
return 0;
}

function set_enctype($newEnctype= "application/x-www-form-urlencoded"){
if($newEnctype != "application/x-www-form-urlencoded" &&
$newEnctype != "multipart/form-data"){
$newEnctype= "application/x-www-form-urlencoded";
}
$this->_enctype=$newEnctype;
return 1;
}

function set_element($key= "",$val= ""){
if(is_array($key)){
$len=sizeof($key);
reset($key);
for($i=0;$i<$len;$i++){
$cur=each($key);
$k=$cur[ "key"];
$v=$cur[ "value"];
$this->_element[$k]=$v;
}
}
else{
if(strlen($key)>0)$this->_element[$key]=$val;
}
return 1;
}

function set_timeout($newTimeout=20){
```

```
$newTimeout=intval($newTimeout);
if($newTimeout<1)$newTimeout=1;
$this->_timeout=$newTimeout;
return 1;
}

function show_post(){
$str= "";
$str.= "Action:".$this->_action. "<br>";
$str.= "Server:".$this->_server. "<br>";
$str.= "Port:".$this->_port. "<br>";
$str.= "File:".$this->_file. "<br>";
$str.= "Enctype:".$this->_enctype. "<br>";

echo $str;

$len=sizeof($this->_element);
reset($this->_element);
for($i=0;$i<$len;$i++){
$cur=each($this->_element);
$key=$cur[ "key"];
$val=$cur[ "value"];
echo "Field:$key = $val<br>\n";
}
return 1;
}

function send($display=0){
// open socket to server
$errno=$errstr=$retstr= "";
$sk = fsockopen($this->_server,
$this->_port,
&$errno,
&$errstr,
$this->_timeout
);
if(!$sk){
return 0;
}
else{
$boundary= "----".md5(uniqid(rand())). "----";
$message=$this->_get_message($boundary);
$str= "";
$str.=strtoupper($this->_method). " ";
$str.=$this->_file. " HTTP/1.0 \r\n";
$str.= "Referer: \r\n";
$str.= "User-Agent: php-HTTP_POST/1.0 \r\n";
$str.= "Host: ".$this->_server. "\r\n";

$str.= "Content-type: ".$this->_enctype;
if($this->_enctype== "multipart/form-data"){
$str.= "; boundary=".$boundary;
}
$str.= " \r\n";

$str.= "Content-length: ".strlen($message). "\r\n\r\n";
$str.=$message;
```

```
fputs($sk,$str);

while(!feof($sk)){
$resp=fgets($sk,80);
$retstr.=$resp;
if($display)echo $resp;
}

fclose($sk);
return $retstr;
}
}

function _get_message($boundary= ""){
$retstr= "";

$len=sizeof($this->_element);
reset($this->_element);

$switch=($this->_enctype== "multipart/form-data")?0:1;

for($i=0;$i<$len;$i++){
$cur=each($this->_element);
$key=$cur[ "key"];
$val=$cur[ "value"];

if($switch){
if(strlen($retstr)!=0)$retstr.= "&";
$retstr.=rawurlencode($key). "=";
$retstr.=rawurlencode($val);
}
else{
$retstr.=$boundary. "\r\n";
$retstr.= "Content-Disposition: form-data; ";
$retstr.= "name=\"$key\"\r\n\r\n$val\r\n\r\n";
}
}
if(!$switch)$retstr.=$boundary. "\r\n";
return $retstr;
}
}

?>
```

La clase utiliza las funciones de networking de php para conectarse al servidor indicado y enviar usando método post los datos que se desean.

Capítulo 16: XML

Una de las tecnologías más necesarias hoy en dia y en el futuro en la mayoría de los portales de Internet que manejan información es XML. XML basado en SGML al igual que HTML es un lenguaje que permite definir otros lenguajes de tipo Mark-UP como por ejemplo HTML. El objetivo de XML es proveer a la Web de un lenguaje standard fuertemente estructurado que permita estructurar contenidos. En XML los documentos pueden ser validados contra un "DTD" (document type definition) para verificar si cumplen lo que dicho DTD determina. El intercambio de DTDs entre distintas instituciones es un buen medio de intercambiar informacion y poder validarla.

PHP provee funciones que permiten parsear documentos XML mediante la biblioteca "Expat" que puede conseguirse libremente en caso de ser necesario, aquellos que usen Apache como Web Server ya tienen expat en forma default.

PHP provee mediante expat un parser XML que puede configurarse para usar "handlers" específicos que manejen los distintos tipos de tags XML, para configurar handlers para los distintos tipos de tags se utilizan las siguientes funciones:

Supported XML handlers

PHP function to set handler	Event description
xml_set_element_handler()	Un evento de elemento se es levantado por expat cada vez que se encuentra un tag de apertura o cierre. Esta función permite setear handlers que se ejecuten cuando ocurren dichos eventos.
xml_set_character_data_handler()	Todo el contenido que no sea de markup de un xml incluyendo espacios en blanco y saltos de línea entre tags es considerado character_data. Es responsabilidad de la aplicación tratar estos datos y para ello es posible setear un handler que se invoque cada vez que se encuentren estos datos.
xml_set_processing_instruction_handler()	XM soporta el uso de "processing instructions" por ejemplo <?php?> es una, cada vez que el parser expat encuentra una PI se invoca esl handler seteado por esta función. Las PIs de tipo <?XML?> no disparan eventos porque no los maneja la aplicación de acuerdo al standard XML.
xml_set_default_handler()	Todo aquello que no corresponda a ningún otro handler disparará un evento deafult que puede capturarse con esta función.
xml_set_unparsed_entity_decl_handler()	Este handler se llama cuando aparecen declaraciones de entidades no parseables (NDATA).
xml_set_notation_decl_handler()	Handler que se invoca cuando se declara una notación XML.
xml_set_external_entity_ref_handler()	Esta función permite setear un handler a invocar cuando el parser xml encuentra una entidad parseable en forma externa con referencia a un file o URL

Notar que las funciones que describimos arriba no son los handlers sino las funciones que permiten definir los handlers a asociar con cada eventos (son funciones de binding)

XML Handlers:

Element Handlers:

xml_set_element_handler (parser_handler, string startElementHandler, string endElementHandler)

La función xml_set_element_handler recibe como parámetros un parser_xml (creado con xml_parser_create) y el nombre de dos funciones que serán los handlers a aplicar cuando se encuentren tags XML, la función necesita el nombre de dos funciones, una para cuando el tag xml abre y otra para cuando el tag xml cierra.

La función startElementHandler debe tener el siguiente prototipo:

boolean=element_start_handler(parser_handler,string_tag,attribs)
boolean=element_end_handler(parser_handler,string_tag)

La función recibe el tag en cuestión y si es un tag que abre recibe los parámetros del tag en caso de estar presentes en un vector asociativo como tercer parámetro. Un ejemplo de tag con atributos es:

<INFO code=”1234” name=”foo”>

En cuyo caso el nombre del tag es “INFO” y además se recibe un asociativo de la forma:
(“code”=>”1234”, “name”=>” foo”)

Como es de esperar los handlers para tags que cierran no reciben atributos.

Se supone que las funciones devuelve true si el parser_handler corresponde a un parser y false en caso contrario.

Character Data Handlers:

xml_set_character_data_handler (parser_handler, string Handler)

La función debe tener como prototipo:

boolean=charhandler(parser_handler, string_data)

Se supone que la función devuelve true si el parser_handler corresponde a un parser y false en caso contrario.

Processing Instructions Handlers:

xml_set_processing_instruction_handler (parser_handler, string_handler_name)

Un processing instruction en XML tiene la forma:

<?target data?>

Ejemplo <?php codigo?> Donde el target es PHP y el data es el código a ejecutar.

El handler responde al prototipo:

handler (parser_hanlder, string_target, string_data)

Default Handler:

xml_set_default_handler (parser_handler, string_handler_name)

Y el handler es de la forma:

handler (parser_hanlder, string_data)

Creación de un Parser XML

Para crear un parser xml se deb usar:

xml_handler=xml_parser_create();

Que crea un parser xml expat y devuelve un handler al mismo.

Parsing:

La función de parsing se invoca usando:

boolean=xml_parse(parser_handler, string_data, boolean);

La función recibe un handler a un parser creado y datos a parsear, se puede llamar a la función “n” veces con distintos pedazos de datos para que la función vaya parseando el documento de a partes.
El tercer parámetro es opcional , si es true indica que los datos que se pasan son los últimos a parsear.
Devuelve true si no hay problemas y false si se produce un error.

Una vez que se termina de parsear un documento es necesario desalocar el parser instanciado usando:

xml_parser_free(parser_hadler);

Uso de Clases:

A veces es conveniente definir que los handlers de un parser_xml sean métodos de una clase o bien crear una clase para parsear determinado tipo de documento XML. Tal vez sea una buena idea definir una clase base para parsear XML en general y luego clases derivadas para los distintos tipos de documentos XML que manejamos. Los nombres que se pasan a las funciones que setean los handlers no prevén construcciones usando objetos por lo que se creo una función especial xml_set_object que le indica al parser xml que los nombres de los handlers registrados son métodos de un objeto determinado:

xml_set_object(xml_handler, referencia_a_un_objeto);

Por ejemplo si estamos dentro de una clase que parsea xml:

xml_set_object($this->xml_parser, &$this);

Ejemplo 1:

Veamos un ejemplo sobre como usar estos handlers para setear manejadores para tags XML, como crear un parser de XML y como parsear el documento, en el ejemplo convertimos XML en HTML:

```
 1
 2 $file = "data.xml";
 3 $map_array = array(
 4     "BOLD"     => "B",
 5     "EMPHASIS" => "I",
 6     "LITERAL"  => "TT"
 7 );
 8
 9 function startElement($parser, $name, $attrs) {
10     global $map_array;
11     if ($htmltag = $map_array[$name]) {
12         print "<$htmltag>";
```

```
13     }
14 }
15
16 function endElement($parser, $name) {
17     global $map_array;
18     if ($htmltag = $map_array[$name]) {
19         print "</$htmltag>";
20     }
21 }
22
23 function characterData($parser, $data) {
24     print $data;
25 }
26
27 $xml_parser = xml_parser_create();
28 // use case-folding so we are sure to find the tag in $map_array
29 xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
30 xml_set_element_handler($xml_parser, "startElement", "endElement");
31 xml_set_character_data_handler($xml_parser, "characterData");
32 if (!($fp = fopen($file, "r"))) {
33     die("could not open XML input");
34 }
35
36 while ($data = fread($fp, 4096)) {
37     if (!xml_parse($xml_parser, $data, feof($fp))) {
38         die(sprintf("XML error: %s at line %d",
39                     xml_error_string(xml_get_error_code($xml_parser)),
40                     xml_get_current_line_number($xml_parser)));
41     }
42 }
43 xml_parser_free($xml_parser);
44
```

Si el documento XML fuera:

Example 5. xmltest2.xml

```
 1
 2 <?xml version="1.0"?>
 3 <foo>
 4 <bold>Este es el titulo</bold>
 6 <emphasis>Descubren sopa que cura la gripe</emphasis>
 7 <literal>Una nueva sopa de <bold>cebolla</bold> que curaria
 8 la gripe fue descubierta hoy en la ciudad de Bolivar, provincia
 9 de Buenos Aires.</literal>
10 </foo>
11
```

El Browser recibiria el siguiente documento:

Example 5. xmltest2.xml

```
  1
  2
  3 <b>Este es el titulo</b>
  6 <i>Descubren sopa que cura la gripe</i>
  7 <tt>Una nueva sopa de <b>cebolla</b> que curaria
  8 la gripe fue descubierta hoy en la ciudad de Bolivar, provincia
  9 de Buenos Aires.</tt>
 10 </foo>
 11
```

Ejemplo 2:

En el siguiente ejemplo vamos a ver una clase definida para parsear noticias en XML en un cierto formato predefinido y mostrarlas en una página convirtiéndolas en HTML, vamos a ver como se usa Orientación a Objetos para encapsular el parser en una sola clase:

El formato de documentos XML que parsea la clase es:

```
<?xml version="1.0" standalone="no"?>
<!DOCTYPE NewsBoy SYSTEM "NewsBoy.dtd">

<NewsBoy>

<story>
        <date>03/31/2000</date>
        <slug>Sooo Busy !</slug>
        <text>
        I haven't posted anything here for a while now as I have been
busy with work (have to pay those bills!). <newline></newline>
        I have just finished a neat little script that stores a complete
record set in a session variable after <newline></newline>
        doing an SQL query. The neat part is that an XML doc is stored in
the session variable an when paging <newline></newline>
        through the results (often near 1000!) the script displays 50
results at a time from the XML doc in the <newline></newline>
        session variable instead of doing another query against the
database. It takes a BIG load off of the <newline></newline>
        database server.
        </text>
</story>

<story>
        <date>03/25/2000</date>
        <slug>NewsBoy Class</slug>
        <text>
        Converted Newsboy to a PHP class to allow better abstraction (as
far as PHP allows.)   <newline></newline>
        Guess that means this is version 0.02 ?!
                        <newline></newline>
        Newsboy will have a section of it's own soon on how to use and
customize the class.    <newline></newline>
        </text>
```

```
</story>

<story>
        <date>03/24/2000</date>
        <slug>NewsBoy is up!</slug>
        <text>
        I have just finished NewsBoy v0.01 !!! <newline></newline>
        It looks quite promising. You may ask, &quot;What the heck is
it?!&quot;. <newline></newline>
        Well it's a simple news system for web-sites, written in PHP,
that makes use of XML for <newline></newline>
        the news data format allowing easy updating and portability
between platforms.
        It uses the built in expat parser for Apache.
        This is just the very first version and there will be loads of
improvements as the
        project progresses.
        </text>
</story>

<story>
        <date>03/24/2000</date>
        <slug>Romeo must Die</slug>
        <text>
        Saw a really cool movie today at Mann called 'Romeo must Die'
<newline></newline>
        Nice fight scenes for a typical kung-fu movie with some 'Matrix'
style effects. <newline></newline>
        One particular cool effect was the 'X-Ray Vision' effect that
occured in various fight scenes. <newline></newline>
        The hero, played by Jet Li, strikes a bad guy and you can see the
bone in his arm crack, in X-RAY vision. <newline></newline>
        There were some funny scenes too when Jet has to play American
football with the bad guys. <newline></newline>
        The official website for the movie is &lt;A
HREF='http://www.romeo-must-die.com'&gt; here &lt;/A&gt;
<newline></newline>
        <newline></newline>
        </text>
        &lt;IMG
SRC=&quot;http://a1996.g.akamaitech.net/7/1996/25/e586077a88e7a4/romeomus
tdie.net/images/image15.jpg&quot; WIDTH=300 &gt;
</story>

</newsboy>
```

El código de la clase es:

```
<?php
/*
NewsBoy: a News system for the web written in PHP by Justin Grant
(Web: jusgrant.cjb.net or justin.host.za.net Mail: justin@glendale.net)
25 March V0.0.2 Converted Newsboy to a PHP class, allowing the layout to be easily modified.
Also added made the HTML that is genrated a little easier to read.
24 March V0.0.1 Just completed the intial version, very rough and basic.
*/

class newsboy {

var $xml_parser;
var $xml_file;
var $html;
var $open_tag ;
var $close_tag ;

//Class Constructor
function newsboy() {
$this->xml_parser = "";
$this->xml_file = "";
$this->html = "";
$this->open_tag = array( //these are the default settings but they are quite easy to modify
"NEWSBOY" => "\n<!-- XML Starts here -->\n<TABLE COLS=1 CELLPADDING=5>",
"STORY" => "<TR><TD BGCOLOR=#222222>",
"DATE" => "<FONT COLOR=#BBBB00>",
"SLUG" => "<FONT COLOR=#00AACC><B>  ",
"TEXT" => "<FONT COLOR=#AAAAAA>",
"PIC" => "",
"NEWLINE" => ""
);
$this->close_tag = array(
"NEWSBOY" => "</TABLE>\n<!-- XML Ends here -->\n\n",
"STORY" => "</TD></TR>",
"DATE" => "</FONT>",
"SLUG" => "</B></FONT><BR>",
"TEXT" => "</FONT>\n",
"PIC" => "<IMAGE SRC=",
"NEWLINE" => "<BR>"
);

}
//Class Destructor (has to be invoked manually as PHP does not support destructors)
function destroy() {
xml_parser_free($this->xml_parser);
}

//Class Members
function concat($str) {
$this->html .= $str;
}

function startElement($parser, $name, $attrs) {
//global $open_tag;
```

```
if ($format= $this->open_tag[$name]) {
$this->html .= $format;
}
}

function endElement($parser, $name) {
global $close_tag;

if ($format= $this->close_tag[$name]) {
$this->html .= $format;
}
}

function characterData($parser, $data) {
$this->html .= $data;
}

/*
function PIHandler($parser, $target, $data) {
//switch ( strtolower($target)){
// case "php":
eval($data);
// break;
//}
}
*/

function parse() {

$this->xml_parser = xml_parser_create();
xml_set_object($this->xml_parser, &$this );
// use case-folding so we are sure to find the tag in $map_array
xml_parser_set_option($this->xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($this->xml_parser, "startElement", "endElement" );
xml_set_character_data_handler($this->xml_parser, "characterData" );
//xml_set_processing_instruction_handler($this->xml_parser, "PIHandler");

if (!($fp = fopen($this->xml_file, "r"))) {
die( "could not open XML input");
}

while ($data = fread($fp, 4096)) {
if (!xml_parse($this->xml_parser, $data, feof($fp))) {
die(sprintf( "XML error: %s at line %d",
xml_error_string(xml_get_error_code($this->xml_parser)),
xml_get_current_line_number($this->xml_parser)));
}
}
}
}
?>
```

En primer lugar se definen los hanlders startElement y closeElement en donde se cuenta con un array asociativo que relaciona el tag xml con el código html a incluir, como vemos se verifica si existe el nombre del tag como clave del vector asociativo (if $clave=$vector[$clave]) si el tag existe entonces se recupera el código a incluir y se appendea al data_member "html" del objeto que será el código html resultante.

El character_data_handler simplemente agrega los datos que recibe al codigo de salida.

El método principal es "parse()", allí se crea un parser xml y se guarda el handler en $this->xml_parser, luego se usa xml_set_object para indicar que las funciones que registraremos como handlers son métodos de este objeto de la forma **xml_set_object($this->xml_parser, &$this)** luego se utiliza xml_parser_set_option para indicar que el parser es case-insensitive. Luego con xml_set_element_handler, xml_set_character_handler se registran los métodos a usar en el xml_parser.

Luego se parsea el file xml pasado al objeto usando una lectura simple de archivos en bloques de 4Kb, usando la función feof como tercer parametro de xml_parse para saber si quedan o no más datos en el archivo.

La forma de usar esta clase es muy sencilla:

$news = new newsboy();
$news->xml_file = "xml/mynews.xml";

o bien:

$news->xml_file = "http://xmldocs.mysite.com/mynews.xml"

Luego llamamos al método parse del objeto:

$news->parse();

Y tomamos el resultado y por ejemplo lo entregamos al browser:

print ($news->html);

Por ultimo destruimos el objeto:

$news->destroy();

Uso de PHP desde XML:

Usando el processing_instruction_handler podemos combinar incluir código PHP dentro de un documento XML, por ejemplo:

```
function processphp($parser,$target,$data) {
  if($target==”php”) {
    eval($data);
  }
}

xml_set_processing_instruction_handler($xml_parser,"processphp");
```

La función "eval" recibe un string con código php y lo ejecuta, como podemos ver con este esquema podemos tener un xml de la forma:

```
<?xml version="1.0"?>
<foo>
<titulo>Descubren sopa curadora en Catelar</titulo>
```

```
<copete>Una sopa con propiedades milagrosas fue descubierta hoy</copete>
<?php print("que cosa!");?>
<cuerpo>La sopa de cebollas del padre Enrico de la iglesia de Castelar
ademas de ser muy sabrosa tiene propiedades curadoras</cuerpo>
</foo>
```

Lo cual nos da opciones muy poderosas de definición y estructuración de contenidos usando XML más toda la posibilidad de programar usando php.

Uso de Coccon con PHP

XML es un lenguaje pensado para estructurar contenidos definiendo lenguajes de marcación, validar documentos y proveer un marco de intercambio de documentos y contenidos. Para poder usar XML desde un Web-Site es necesario definir alguna tecnología que permita transformar los contenidos definidos en XML en "presentación" típicamente usando HTML. Para transformar XML en html pueden usarse las funciones de parsing que estudiamos en php. Sin embargo existe un lenguaje creado específicamente para manjar transformaciones de documentos XML denominado XSL, XSL es un lenguaje de transformación de documentos XML en otros documentos XML, como es posible considerar HTML como un sub-set de XML es posible transformar XML en HTML usando XSL, mediante el uso de XSL es posible definir otras transformaciones para el mismo documento XML como por ejemplo un dispositivo WAP u otros.

Coccon es un proyecto de la Apache Software Foundation que provee un conjunto de Servlets que se integran con el Apache Web Server para maipular documentos XML, una de las cosas que Cocoon maneja es precisamente las transformaciones XSL.

La mejor estrategia de integración entre XSL y PHP mientras que PHP no soporte nativamente XSL es el uso de Cocoon, habitualmente se puede desde PHP generar un documento XML usando clases de generación de XML a partir de datos en la base de datos u otros formatos y luego generar un archivo XML, a continuación solo queda redireccionar al browser al archivo XML generado para que Coccon entre en acción parseando el documento XML y realizando la transformación que se desee usando la plantilla XSL seleccionada. Este marco de trabajo si bien es muy nuevo actualmente como para que podamos aportar datos sobre su funcionamiento puede ser uno de los caminos más acertados para integrar PHP, XML, XSL y HTML separando adecuadamente Aplicaciones, Contenidos y Presentación de los mismos.

Capítulo 17: Manejo de Mail en PHP

Conexión a un server IMAP o POP3:

mail_handler=imap_open(string_mbox,user,password);

Donde mbox es de la forma:

{IP:PORT}MailBox

Ejemplos:

$mail=imap_open(“{190.190.190.190:143}INBOX”,”user”,”pass”);

Conexión a la carpeta INBOX de un servidor IMAP (puerto 143)

$mail=imap_open(“{190.190.190.190:110}”,”user”,”pass”);

Conexión a la carpeta raíz de un servidor POP3 (puerto 110)

Una vez establecida la conexión la función devuelve un handler que se utiliza en el resto de las funciones para acceder a las carpetas y mails dentro de las mismas.

Ejemplo:

Example 1. imap_open() example

```
 1
 2 $mbox = imap_open ("{your.imap.host:143}", "username", "password");
 3
 4 echo "<p><h1>Mailboxes</h1>\n";
 5 $folders = imap_listmailbox ($mbox, "{your.imap.host:143}", "*");
 6
 7 if ($folders == false) {
 8     echo "Call failed<br>\n";
 9 } else {
10     while (list ($key, $val) = each ($folders)) {
11         echo $val."<br>\n";
12     }
13 }
14
15 echo "<p><h1>Headers in INBOX</h1>\n";
16 $headers = imap_headers ($mbox);
17
18 if ($headers == false) {
19     echo "Call failed<br>\n";
20 } else {
21     while (list ($key,$val) = each ($headers)) {
22         echo $val."<br>\n";
23     }
24 }
25
26 imap_close($mbox);
27
```

Una vez terminada la conexión se usa:

imap_close(mail_handler);

Manejo de MailBoxes:

int=imap_createmailbox (mail_handler string_mbox)

String mbox debe estar codificado con imap_utf7_encode() y el formato del string es el mismo que en imap_open.

Ejemplo:

Example 1. imap_createmailbox() example

```
 1
 2 $mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
 3      || die("can't connect: ".imap_last_error());
 4
 5 $name1 = "phpnewbox";
 6 $name2 = imap_utf7_encode("phpnewböx");
 7
 8 $newname = $name1;
 9
10 echo "Newname will be '$name1'<br>\n";
11
12 # we will now create a new mailbox "phptestbox" in your inbox folder,
13 # check its status after creation and finaly remove it to restore
14 # your inbox to its initial state
15 if(@imap_createmailbox($mbox,imap_utf7_encode("{your.imap.host}INBOX.$newname")))   {
16   $status = @imap_status($mbox,"{your.imap.host}INBOX.$newname",SA_ALL);
17   if($status) {
18     print("your new mailbox '$name1' has the following status:<br>\n");
19     print("Messages:   ". $status->messages   )."<br>\n";
20     print("Recent:     ". $status->recent     )."<br>\n";
21     print("Unseen:     ". $status->unseen     )."<br>\n";
22     print("UIDnext:    ". $status->uidnext    )."<br>\n";
23     print("UIDvalidity:". $status->uidvalidity)."<br>\n";
24
25
if(imap_renamemailbox($mbox,"{your.imap.host}INBOX.$newname","{your.imap.host}INBOX.$name2")) {
26       echo "renamed new mailbox from '$name1' to '$name2'<br>\n";
27       $newname=$name2;
28     } else {
29       print "imap_renamemailbox on new mailbox failed: ".imap_last_error()."<br>\n";
30     }
31   } else {
32     print  "imap_status on new mailbox failed: ".imap_last_error()."<br>\n";
33   }
34   if(@imap_deletemailbox($mbox,"{your.imap.host}INBOX.$newname")) {
35     print "new mailbox removed to restore initial state<br>\n";
36   } else {
37                         print         "imap_deletemailbox     on     new     mailbox     failed:
".implode("<br>\n",imap_errors())."<br>\n";
38   }
39
40 } else {
41   print  "could not create new mailbox: ".implode("<br>\n",imap_errors())."<br>\n";
42 }
43
44 imap_close($mbox);
45
```

Devuelve true si pudo crear el mailbox o false en caso contrario.

int=imap_deletemailbox (mail_handler, string_mbox);

Elimina el mailbox indicado, el formato de mbox es el mismo que en imap_open.

int=imap_renamemailbox (mail_handler, string_old_mbox, string_new_mbox)

Permite renombrar un mailbox, el nombre del mailbox debe estar en el mismo formato que en imap_open.

obj_array=imap_getmailboxes (mail_stream, string_ref, string_pattern)

Devuelve un vector de objetos con información sobre los mailboxes

Los objetos que se encuentran en el vector tienen seteados los siguientes data_members:

- name – Nombre del mailbox (completo) encodeado, decodificar con imap_utf7_decode()
- delimiter – Delimitador usado para separar la jerarquía de mailboxes
- attributes – Es un bitmask que puede compararse con:
  - LATT_NOINFERIORS (el mailbox no tiene subcarpetas)
  - LATT_NOSELECT (es un mailbox no seleccionable)
  - LATT_MARKED (mailbox marcado)
  - LATT_UNMARKED (mailbox no marcado)

Ejemplo:

Example 1. imap_getmailboxes() example

```
 1
 2 $mbox = imap_open("{your.imap.host}","username","password")
 3       || die("can't connect: ".imap_last_error());
 4
 5 $list = imap_getmailboxes($mbox,"{your.imap.host}","*");
 6 if(is_array($list)) {
 7   reset($list);
 8   while (list($key, $val) = each($list))
 9   {
10     print "($key) ";
11     print imap_utf7_decode($val->name).",";
12     print "'".$val->delimiter."',";
13     print $val->attributes."<br>\n";
14   }
15 } else
16   print "imap_getmailboxes failed: ".imap_last_error()."\n";
17
18 imap_close($mbox);
19
```

object=imap_status (mail_handler, string_mailbox, SA_ALL)

SA_ALL es una constante para recuperar toda la información sobre el mailbox, devuelve un objeto con los siguientes data members seteados:

- messages – número de mensajes en el mailbox
- recent – número de mensajes recientes en el mailbox
- unseen – número de mensajes no vistos en el mailbox

Ejemplo:

Example 1. imap_status() example

```
  1
  2 $mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
  3       || die("can't connect: ".imap_last_error());
  4
  5 $status = imap_status($mbox,"{your.imap.host}INBOX",SA_ALL);
  6 if($status) {
  7   print("Messages:   ". $status->messages   )."<br>\n";
  8   print("Recent:     ". $status->recent     )."<br>\n";
  9   print("Unseen:     ". $status->unseen     )."<br>\n";
 10   print("UIDnext:    ". $status->uidnext    )."<br>\n";
 11   print("UIDvalidity:". $status->uidvalidity)."<br>\n";
 12 } else
 13   print "imap_status failed: ".imap_lasterror()."\n";
 14
 15 imap_close($mbox);
 16
```

int imap_num_msg (mail_handler)

Devuelve el número de mensajes en el mailbox actual. (El abierto por el mail_handler)

int imap_num_recent (mail_handler)

Devuelve el número de mensajes recientes del mailbox correspondiente a mail_handler.

Manejo de mensajes:

object=imap_fetchstructure (mail_handler, int msg_number)

Devuelve un objeto con la estructura del mensaje recuperado:

Table 1. Returned Objects for imap_fetchstructure()

type	Primary body type
encoding	Body transfer encoding
ifsubtype	True if there is a subtype string
subtype	MIME subtype
ifdescription	True if there is a description string
description	Content description string
ifid	True if there is an identification string
id	Identification string
lines	Number of lines
bytes	Number of bytes
ifdisposition	True if there is a disposition string
disposition	Disposition string
ifdparameters	True if the dparameters array exists
dparameters	Disposition parameter array
ifparameters	True if the parameters array exists
parameters	MIME parameters array
parts	Array of objects describing each message part

Cuando el mensaje es multipart “parts” es un vector donde cada elemento es un objeto con los siguientes datamembers:

- type
- encoding
- subtype
- description
- lines
- disposition

Luego según el trasnfer encoding (ver tabla 3) se puede usar la función de decodificación apropiada

Table 2. Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Table 3. Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

Las funciones de decodificación provistas son:

string=imap_base64(string) convierte de base 64 a 8 bits
string=imap_8bit(string) convierte de 8 bits a quoted printable
string=imap_utf7_decode(string) convierte de 7 bits a 8 bits
string=imap_qprint(string) convierte de quoted printable a 8 bits
string=imap_binary(string) convierte de 8 bits a base64

El formato de salida “string” es 8 bits, si el formato de encoding es otro basta con usar la función apropiada.

string=imap_fetchbody (mail_handler, int msg_number, string part_number)

Recupera la parte indicada del body de un determinado mensaje. No realiza ningún tipo de decodificación.

array= imap_headers (mail_handlers)

Devuelve un vector de headers para el mailbox actual (cada header es un string y es un elemento del vector)

object=imap_rfc822_parse_headers(string headers)

Parsea un header de acuerdo a rfc822, devuelve un objeto con los siguientes data_members:

- remail
- date
- Date
- subject
- Subject
- in_reply_to
- message_id
- newsgroups
- followup_to
- references

string imap_body (mail_handler, int msg_number)

Devuelve el body de un determinado mensaje.

Envío de mail:

Enviar mail desde PHP es sencillo con la función:

bool= mail (string to, string subject, string message [, string additional_headers])

Example 1. Sending mail.

```
  1
  2 mail("rasmus@lerdorf.on.ca", "My Subject", "Line 1\nLine 2\nLine 3");
  3
```

Example 2. Sending mail with extra headers.

```
  1
  2 mail("nobody@aol.com", "the subject", $message,
   3       "From: webmaster@$SERVER_NAME\nReply-To: webmaster@$SERVER_NAME\nX-
Mailer: PHP/" . phpversion());
  4
```

Capítulo 18: Networking y FTP usando PHP

Funciones de FTP:

Uno de los problemas no solucionados por la mayoría de los lenguajes de scripting para la web es la transferencia de archivos entre diversos sites, usando NFS (Network file system) es posible crear zonas de un file-system que sean compartidas por más de un site, pero esto obliga a usar NFS, a tener una sobrecarga administrativa y además insertar varios problemas de seguridad delicados relacionados con NFS. El método más seguro y confiable para transferir archivos, PHP provee funciones nativas para usar FTP en forma directa desde un script php, las funciones más usadas son las siguientes:

ftp_handler=ftp_connect(host);

Ejemplo

$ftp=ftp_connect("100.100.100.100");

Si el puerto de FTP no es el default (21) se puede pasar como segundo parámetro de ftp_connect, la función devuelve un handler a ser usado por el resto de las funciones de FTP.
Devuelve true/false segun el éxito o fracaso de la conexión.

int=ftp_login(ftp_handler,string_user,string_password);

Realiza un login a la conexión FTP indicada por el handler. (devuelve true/false según el éxito o fracaso del login)

int=ftp_chdir(ftp_handler, string_directory);

Cambia al directorio especificado como segundo parámetro dentro de la conexión ftp abierta para el ftp_handler pasado.Devuelve true/false

int=ftp_get (ftp_handler, string_local_file, string_remote_file, int mode);

Transfiere el archivo string_remote_file a la maquina donde corre el script creandolo con el path indicado en string_local_file, el modo debe ser una constante que puede ser FTP_BINARY o FTP_ASCII

Ejemplo

$ret=ftp_get($ftp,"/temp/cosa.dat","cosa.dat",FTP_BINARY);

Notar que no deben usarse comillas en el cuarto parámetro por tratarse de una constante y no de un string.

int=ftp_fget (ftp_handler, file_handler, string_remote_file, int mode)

Idem anterior pero el segundo parámetro es un handler a un archivo abierto con fopen en donde se guardaran los datos leídos desde la conexión ftp.

int=ftp_put (ftp_handler, string_remote_file, string_local_file, int mode)

Transfiere un archivo local indicado por string_local_file mediante la conexión ftp abierta creando un archivo de nombre string_remote_file.

int=ftp_fput (ftp_handler, string_remote_file, file_handler, int mode)

Idem anterior pero el archivo a trasnferir se indica por el file_handler de un archivo abierto con fopen.

string=ftp_mkdir (ftp_handler, string_directory)

Crea un directorio en la conexión ftp abierta, devuelve el nombre del directorio creado o falso si no pudo crearlo.

int=ftp_rmdir (ftp_handler, string directory)

Elimina el directorio indicado, devuelve true/false.

int=ftp_cdup (ftp_handler)

Cambia al directorio padre del actual.

array=ftp_rawlist (ftp_handler, string_directory)

Devuelve un vector con la lista de archivos presentes en el directorio indicado de acuerdo al comando FTP ftp_list, el resultado es un vector donde cada elemento del vector es una línea devuelta por ftp_list.

int=ftp_size (ftp_handler, string_remote_file)

Devuelve el tamaño del archivo indicado.

ftp_quit(ftp_handler);

Se desconecta de la conexión ftp realizada con ftp_connect.

Networking en PHP:

PHP dispone de varias funciones de networking la más usada y la más flexible es fsockopen que permite conectarse a un socket en un host determinado por una dirección IP y un puerto, mediante esta funcion es posible conectarse a servidores HTPP, FTP, Telnet, IMAP, POP3 y otros protocolos.

Es de destacar que la funcionalidad de Networking de PHP es como CLIENTE, PHP no puede crear un socket con nombre y hacer un "listen" de conexiones a dicho port por lo que no puede funcionar como servidor.

La sintaxis de fsockopen es:

file_handler=fsockopen (string_hostname, int port , int errno , string_errstr , double timeout)

Los tres últimos parámetros son opcionales.
Hostname es el nombre o dirección IP del host al cual conectarse.
Port es el número de puerto al cual conectarse en el host.
errno debe ser una referencia a una variable en donde se guarda el número de error en caso de no poder conectarse.
errstr es una referencia a una variable en donde se guarda un mensaje de error en caso de no poder conectarse
El timeout es el tiempo máximo a esperar por la conexión en segundos.

Devuelve un file handler o false según pueda o no conectarse. El file hanlder devuelto puede luego usarse como un archivo normal usando fgets, fputs, feof, fclose, etc...

Ejemplo:

```
 1
 2 $fp = fsockopen ("www.php.net", 80, &$errno, &$errstr, 30);
 3 if (!$fp) {
 4     echo "$errstr ($errno)<br>\n";
 5 } else {
 6     fputs ($fp, "GET / HTTP/1.0\n\n");
 7     while (!feof($fp)) {
 8         echo fgets ($fp,128);
 9     }
10     fclose ($fp);
11 }
12
```

En este ejemplo abrimos el puerto 80 (Protocolo HTTP) de un host (www.php.net)
Luego ponemos en el socket un request de HTTP y entramos en un loop recuperando el contenido que devuelve el server. Es un mini simulador de browser HTTP.

Capítulo 19: Funciones adicionales de PHP

Persistencia:

Una de las características importantes en lenguajes orientados a objetos o lenguajes de scripting modernos es la persistencia, un objeto persistente es aquel que puede almacenarse en un medio de almacenamiento secundario (un archivo o una base de datos) para luego recuperarlo. PHP provee de dos funciones que permiten realizar esto serializando y des-serializando variables de PHP.

string=serialize(var);

Recibe cualquier variable de PHP incluso un objeto y devuelve un string que es una representación de la variable, dicho string puede almacenarse en un archivo o una base de datos para lograr persistencia.

var=unserialize(string);

Recibe un string que es la serialización de una variable, des-serializa y asigna a la variable pasada. Para des-serializar un objeto es necesario que el script que usa unserialize disponga de la definición de la clase.

Funciones de hashing y encriptación:

string=md5(string)

Devuelve un string de 32 bytes que es un "digest" del string original, es decir aplica al string original una función de hashing unidireccional.

string=crypt(string)

Encripta un string usando el método unidireccional de Unix, usado por ejemplo para almacenar passwords, el string devuelto es de extensión variable. No se puede desencriptar.

Generación de identificadores únicos:

string=uniqid(string_base);

Construye un identificador único tomando como base un string pasado, por ejemplo para generar identificadores únicos de 32 bits aleatorios se usa:

$better_token = md5 (uniqid (rand()));

Ejecución de codigo PHP:

eval(string_codigo);

Evalúa el string como si fuera código PHP y lo ejecuta.

Ejemplo:

eval("print(\"hola\")");

Imprime hola como si se ejecutara la instrucción print, la función eval es útil en varios casos por ejemplo para guardar código PHP en un archivo o base de datos y luego recuperarlo y ejecutarlo dinámicamente (por ejemplo para que usuarios de un web site suban sus propios scripts PHP) o bien usando funciones de parsing XML para insertar en XML processing-instructions de tipo <?php código ?> y luego en el script php que parsea el XML ejecutar el código php con eval.

Control del script:

sleep(segundos);

Hace una pause de la cantidad de segundos indicados.

die(string);

Termina la ejecución del script imprimiendo el string pasado.

exit();

Finaliza la ejecución del script.

Manejo de URLs

string=base64_decode(string);

Decodifica un string encodeado en base64.

string=base64_encode(string);

Codifica un string a base64.

string=urlencode(string);

Codifica un string de acuerdo a RFC1738 es decir reemplaza todos los caracteres que no sean alfanuméricos o "_","."," por un signo "%" seguido de una representación hexadecimal del caracter de acuerdo a RFC1738, los espacios en blanco se codifican como %20 por ejemplo. Este formato es "seguro" para pasarlo como query_string en un URL

Ejemplo tipico:

```
$ulr_string=urlencode($string_raro);
print("<a href=\"http://algo.com?$url_string\">");
etc...
```

string=urldecode(string);

Decodifica un string encodeado con urlencode.

array=parse_url(string_url);

Recibe un string representando una URL y devuelve un vector asociativo de donde pueden recuperarse las sigiuientes claves:

"scheme", "host", "port", "path", "query"

Ejemplo:
http://www.yahoo.com:8080/pruebas/coso.php?hola=5&cosa=6

Scheme : http
Host : www.yahoo.com
Port : 8080
Path : /pruebas/coso.php
Query : hola=5&cosa=6