

Llenguatges de guions de servidor

Eduard García Sacristán, Eduard Latorre Jarque, Montserrat
Madridejos Mora, Raúl Velaz Mayo

Adaptació de continguts: Eduard García Sacristán

Índex

Introducció	5
Resultats d'aprenentatge	7
1 Instal·lació de servidors d'aplicacions web	9
1.1 Anàlisi de requeriments	9
1.1.1 L'URL. Accés a recursos	9
1.1.2 HTTP: comunicació entre servidor i client	10
1.1.3 Contenedors web	11
1.2 Servidors web	12
1.2.1 Servidor web Apache	13
1.2.2 Instal·lació d'Apache	14
1.2.3 Configuració bàsica d'Apache	14
1.2.4 Configuració de directoris personals	15
1.2.5 Configuració de mòduls	16
1.2.6 Control d'accessos	17
1.3 Llenguatges script	18
1.3.1 Llenguatges script de client	18
1.3.2 Llenguatges script de servidor	19
1.3.3 Instal·lació de PHP5	20
1.4 MySQL	20
1.4.1 Instal·lació de MySQL a Debian	21
1.5 phpMyAdmin	21
1.5.1 Instal·lació del phpMyAdmin	22
1.6 Utilitats de prova i instal·lació integrada	23
1.6.1 Utilitats de prova	24
1.7 Documentació	26
1.7.1 Eines d'ajuda a la creació	26
2 Programació web de servidor	29
2.1 El llenguatge PHP	29
2.1.1 Funcionament del llenguatge PHP	29
2.1.2 Configuració del llenguatge PHP	30
2.1.3 Eines de programació per a PHP	31
2.1.4 Sintaxi del llenguatge PHP	31
2.1.5 Tipus de dades del llenguatge PHP	32
2.1.6 Operadors	33
2.1.7 Cadenes i matrius en PHP	36
2.1.8 Variables	39
2.1.9 Estructures de control i sentències	40
2.1.10 Funcions	48
2.1.11 Classes i objectes	49
2.2 Integració PHP amb HTML	50

2.2.1	L'ús del PHP en formularis	51
2.2.2	Tipus de dades en PHP	54
2.2.3	Variables	57
2.2.4	Excepcions en PHP	63
2.2.5	Galetes (cookies)	64
2.2.6	Enciptació de dades amb el PHP	65
2.3	Programació orientada a objectes en PHP	69
2.3.1	Les classes en PHP	70
2.3.2	Objectes en PHP	72
2.3.3	Herència en PHP	74
3	Accés a bases de dades des de PHP	79
3.1	Gestors de bases de dades més usats	79
3.2	Gestió de bases de dades amb PHPMyAdmin	80
3.3	Connexió amb el servidor MySQL	82
3.4	Tancar la connexió a la base de dades	83
3.5	Selecció de la base de dades	84
3.6	Treball amb la base de dades	84
3.6.1	Inserció d'informació	85
3.6.2	Consulta d'informació	86
3.6.3	Modificació d'informació	88
3.6.4	Esborrar registres	89
3.6.5	Gestió d'errors	90
3.7	Creació d'una base de dades	90
3.7.1	Creació de taules	91
3.8	Mecanismes de seguretat	93
3.8.1	Codificació de la informació	93
3.8.2	Accés restringit a pàgines	95
3.8.3	Seguretat a la base de dades	95
3.9	Verificació i proves	97
3.9.1	Nagios	98
3.9.2	Pandora FMS	98
3.9.3	Proves de rendiment	99

Introducció

Actualment és difícil trobar pàgines web que estiguin creades estàticament. Els llenguatges de guions de servidors permeten crear webs d'una forma ràpida i flexible, permetent la interacció amb l'usuari i utilitzant dades provinents de bases de dades. PHP és probablement el llenguatge de guions de servidors més popular actualment i, juntament amb la base de dades MySQL i el servidor web Apache, fan un equip molt potent amb el qual treballar continguts web.

En l'apartat “Instal·lació de servidors web” veureu el funcionament bàsic d'un servidor web i tot el procés des que es fa la petició web fins que els resultats es mostren per pantalla. Apreneu a instal·lar un servidor web, un servidor de bases de dades i el llenguatge de guions de servidor.

En l'apartat “Programació web de servidor” aprendreu a programar en PHP i la seva integració dins del llenguatge de marques HTML. Dins de PHP hi ha una gran varietat de funcions que fareu servir per realitzar els vostres webs. Per últim veureu com fer servir el paradigma d'orientació d'objectes dins dels vostres programes.

En l'apartat “Accés a bases de dades des de PHP” veureu les funcions i mecanismes bàsics per treballar amb dades que es troben a una base de dades dins de programes escrits en PHP. També veureu els diferents mecanismes de seguretat que podeu aplicar als webs dinàmics i com fer verificació i proves del web.

Per seguir els continguts d'aquest mòdul, és convenient anar fent les activitats i els exercicis d'autoavaluació i llegir els annexos que s'indiquin. Tot i que les unitats formatives tenen un contingut important des del punt de vista conceptual, sempre s'ha procurat donar-los un enfocament pràctic en les activitats proposades.

Resultats d'aprenentatge

En finalitzar aquesta unitat, l'alumne/a:

1. Prepara l'entorn de desenvolupament i els servidors d'aplicacions web instal·lant i integrant les funcionalitats necessàries.

- Identifica el programari necessari per al seu funcionament.
- Identifica les diferents tecnologies emprades.
- Instal·la i configura servidors web i de bases de dades.
- Reconeix les possibilitats de processament en els entorns client i servidor.
- Afegeix i configura els components i mòduls necessaris per al processament de codi al servidor.
- Instal·la i configura l'accés a bases de dades.
- Estableix i verifica la seguretat en els accessos al servidor.
- Utilitza plataformes integrades orientades a la prova i desenvolupament d'aplicacions web.
- Documenta els procediments realitzats.

2. Genera documents web utilitzant llenguatges de guions de servidor.

- Identifica els llenguatges de guions de servidor més rellevants.
- Reconeix la relació entre els llenguatges de guions de servidor i els llenguatges de marques utilitzats en els clients.
- Reconeix la sintaxi bàsica d'un llenguatge de guions concret.
- Utilitza estructures de control del llenguatge.
- Defineix i utilitza funcions.
- Utilitza formularis per introduir informació.
- Estableix i utilitza mecanismes per assegurar la persistència de la informació entre diferents documents web relacionats.
- Identifica i assegura els usuaris que accedeixen al document web.
- Verifica l'aïllament de l'entorn específic de cada usuari.

3. Genera documents web amb accés a bases de dades utilitzant llenguatges de guions de servidor.

- Identifica els sistemes gestors de bases de dades més utilitzats en entorns web.
- Verifica la integració dels sistemes gestors de bases de dades amb el llenguatge de guions de servidor.
- Configura en el llenguatge de guions la connexió per a l'accés al sistema gestor de base de dades.
- Crea bases de dades i taules en el gestor utilitzant el llenguatge de guions.
- Obté i actualitza la informació emmagatzemada en bases de dades.
- Aplica criteris de seguretat en l'accés dels usuaris.
- Comprova el funcionament i el rendiment del sistema.

1. Instal·lació de servidors d'aplicacions web

Internet s'ha convertit en la principal eina d'intercanvi d'informació a la societat actual. La connexió dels ordinadors a les xarxes locals i aquestes xarxes a d'altres de grans dimensions ha possibilitat la comunicació global d'informació entre els ordinadors arreu del món. Per poder oferir serveis web, els servidors han de tenir instal·lats una sèrie de serveis. El servidor web permet l'enviament de continguts a altres ordinadors a través de la xarxa. Els preprocessadors d'hipertext (com PHP) permeten crear continguts web de forma dinàmica. Els servidors de bases de dades permeten emmagatzemar informació de forma estructurada que es pot fer servir per generar contingut web. Aquests serveis han de ser instal·lats i configurats correctament abans de començar a crear el contingut web dinàmic del nostre web.

1.1 Anàlisi de requeriments

Un servidor web és una peça de programari que respon a les peticions dels navegadors i lliura la pàgina per al navegador a través d'Internet. Quan es crida a una pàgina web per l'adreça –l'URL (*uniform resource locator*), per exemple, www.ioc.org/index.html–, la comunicació entre el navegador i el servidor és possible gràcies a tres protocols:

- **TCP**(*Transmission Control Protocol*, protocol de control de transmissió): és el responsable de fer que el missatge arribi a la destinació sense errors.
- **IP**(*Internet Protocol*): és el responsable de fer que el missatge trobi el camí fins al servidor.
- **HTTP**(*Hypertext Transfer Protocol*, protocol de transferència d'hipertext): és el protocol que ha indicat l'usuari a l'hora de demanar el recurs al servidor. La primera part d'un recurs URL correspon al protocol que utilitzaran client i servidor per intercanviar dades.

Protocols

El navegador utilitza el protocol HTTP sobre TCP/IP per demanar recursos (URL) als servidors web.

1.1.1 L'URL. Accés a recursos

Una adreça URL té la finalitat d'accedir a un recurs en un servidor. Totes les adreces URL tenen diferents parts; agafem com a exemple `http://ioc.xtec.cat:80/educacio/fp-cicles-formatius/cfgs/index.html`:

- **Protocol**: s'indica abans de `://` i en aquest cas és HTTP. Hi ha altres protocols, com ara FTP o HTTPS.

- **Adreça del servidor:** el lloc d'on es vol obtenir el recurs (ioc.xtec.cat).
- **Port:** el port del servidor destinat a l'aplicació encarregada de gestionar les peticions HTTP (port del servidor web). En cas de que no estigui especificat es fa servir el port estàndard per HTTP (el port 80).
- **Ruta fins al recurs:** correspon a la ruta a la carpeta dins del servidor web on es troba el recurs (educacio/fp-cicles-formatius/cfgs/).
- **Recurs:** és el recurs o fitxer que demanem al servidor (index.html).

El que primer fa el navegador és comunicar-se amb un servidor de noms per obtenir l'adreça IP que correspon al nom de la màquina ioc.xtec.cat. Amb aquesta IP es connecta a la màquina servidor. A continuació, el navegador crea una connexió amb l'adreça IP del servidor al port 80, que és el port utilitzat per defecte en els servidors web. Una vegada el client i el servidor estan connectats, “parlen” fent servir el protocol HTTP per tal que el client indiqui quins recursos vol del servidor. El servidor enviarà aquests recursos al client a través de la connexió establerta.

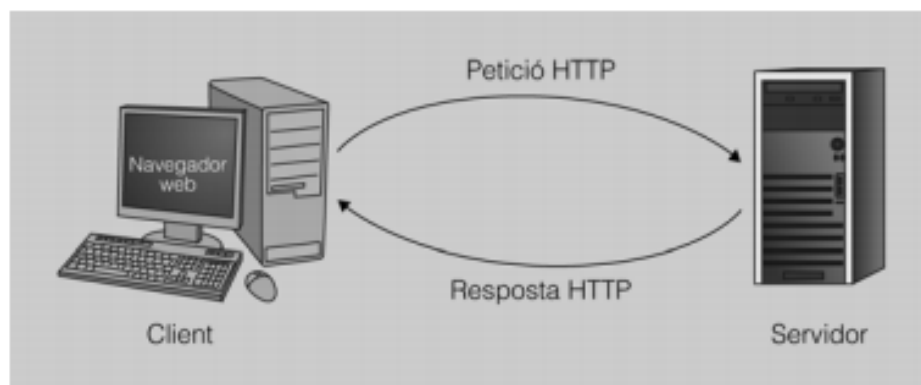
1.1.2 HTTP: comunicació entre servidor i client

Un cop establerta la connexió entra en joc el protocol HTTP: el navegador envia una petició al servidor en què sol·licita el recurs index.html (figura 1.1). El servidor processa aquesta petició i retorna la pàgina sol·licitada al navegador, que interpreta les etiquetes HTML i la presenta a l'usuari.

Sol·licitud client

El client envia, utilitzant el protocol HTTP, una capçalera amb informació sobre el recurs que vol, el mètode que s'ha d'utilitzar. Després de la capçalera, s'envia, si n'hi ha, la informació d'usuari.

FIGURA 1.1. Petició del client d'un recurs del servidor



La informació que el client envia al servidor en la petició és la següent:

- El mètode HTTP: l'acció que s'ha de fer.
- El recurs a què s'ha d'accedir (una part de l'URL).
- La informació que l'usuari envia al servidor.

La informació que el servidor envia en la resposta té dues parts ben diferenciades: la capçalera i el contingut.

- La capçalera conté el codi que indica si la petició s'ha complert. També conté el tipus de contingut que enviarà al client.
- El contingut (text, codi HTML, imatges, etc.) del recurs demanat.

Mètodes GET i POST

La primera part d'una petició HTTP és el tipus de mètode, que serveix per indicar-li el tipus de petició que s'ha fet. Els mètodes poden ser HEAD, TRACE, DELETE, OPTIONS, CONNECT, GET i POST. Aquests dos últims són els que realment utilitzareu.

Tant amb GET com amb POST es pot enviar informació del client al servidor, però hi ha algunes diferències.

La informació enviada amb el mètode GET s'envia després de l'adreça URL, per exemple: *ioc.xtec.cat/index.php?opc=edu&lang=cat*. La informació d'usuari són les dades que hi ha a partir del símbol ?. Aquest fet implica que la quantitat de dades que es poden enviar amb GET estigui limitada. A més a més, la informació que s'envia d'aquesta manera és pública ja que apareix a la pròpia URL, que és fàcilment accessible. Per tant no es pot fer servir per enviar dades privades, com una clau d'accés, per exemple.

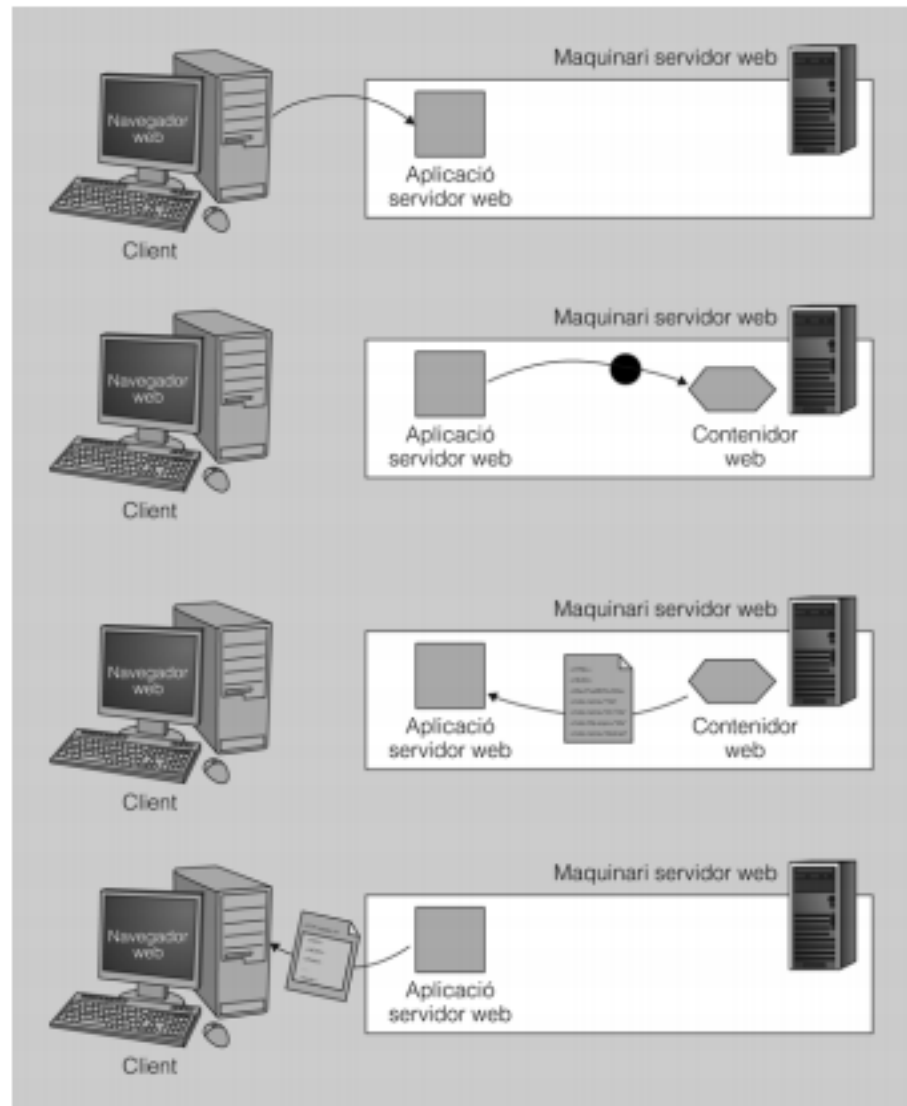
Amb el mètode POST, en canvi, la informació s'envia com a part del contingut i no és visible per a l'usuari.

1.1.3 Contenedors web

El problema dels primers servidors web és que solament entregaven recursos estàtics. Reben una petició, miraven si el recurs estava disponible i l'enviaven.

L'avantatge de les pàgines web dinàmiques és que el contingut es genera dinàmicament i, per tant, no existeix estàtica i prèviament, al servidor.

Els servidors web tan sols saben enviar pàgines estàtiques. Perquè un servidor web pugui treballar amb contingut dinàmic és necessari instal·lar una altra aplicació que ajudi el servidor web a servir contingut dinàmic als clients (com s'observa a la figura 1.2). Les aplicacions encarregades de fer aquesta tasca s'anomenen contenidors web i són les que permeten treballar amb PHP, Java, ASP, C, Python, etc. El seu funcionament és transparent pel client.

FIGURA 1.2. Funcionament d'un servidor web i d'un contenidor**Servidors i contenidors**

Els servidors només saben processar continguts estàtics; per ser capaços de treballar amb continguts dinàmics, com una base de dades, necessiten l'ajuda de programes externs: els contenidors.

En la figura 1.2 podeu observar el funcionament d'un servidor web juntament amb el contenidor. Quan el servidor troba un recurs que no és estàtic, traspasa la petició al contenidor web. Aquest contenidor processa la petició, genera dinàmicament una resposta en forma de contingut web i la passa al servidor. Finalment el servidor envia el contingut al client.

1.2 Servidors web

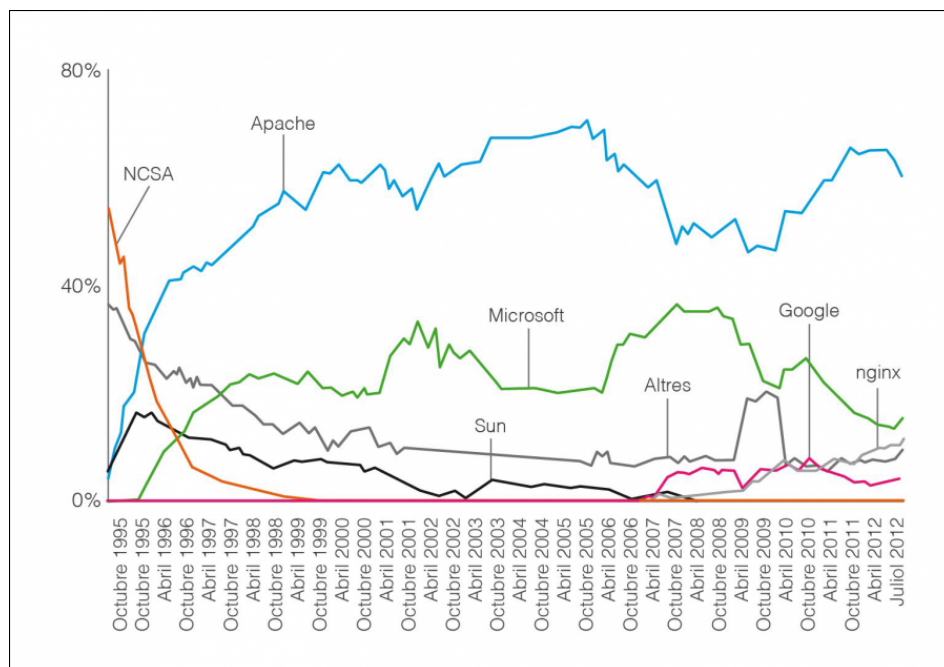
Els servidors web poden estar implementats dins del *kernel* del sistema operatiu o dins de l'espai de memòria d'usuari (com qualsevol altra aplicació del sistema).

Els **servidors web dins del *kernel***, com ara l'IIS de Windows, generalment s'executen més ràpid ja que són part integrant del sistema i poden fer servir tots els recursos de maquinari directament quan els necessiten (com ara la memòria sense paginar o la memòria cau).

Els **servidors web que s'executen en mode d'usuari** han de demanar al sistema quan tenen necessitats de memòria o recursos de CPU. Aquestes peticions tarden en realitzar-se i no sempre poden ser satisfetes, ja que el sistema reserva els recursos pel seu ús i l'ha de repartir entre tota la resta d'aplicacions en execució.

Com es pot veure en la figura 1.3 i la taula 1.1, tot i que avui en dia hi ha molts servidors web diferents, Apache és el servidor web més popular a Internet amb un 65% de quota de mercat. La resta del mercat està repartit entre els servidor d'altres companyies, els més populars dels quals són IIS de Microsoft i nginx.

FIGURA 1.3. Servidors web més populars a Juliol de 2012



Font: <http://news.netcraft.com>

TAULA 1.1. Servidors web més utilitzats per les pàgines web més visitades d'Internet a Juliol de 2012

Desenvolupador	Juny 2012	Percentatge	Juliol 2012	Percentatge	Canvi
Apache	448.452.703	64,33%	409.185.675	61,45%	-2,89
Microsoft	95.891.537	13,76%	97.385.377	14,62%	0,87
nginx	72.881.755	10,46%	73.833.173	11,09%	0,63
Google	22.464.345	3,22%	22.931.169	3,44%	0,22

Font: <http://news.netcraft.com>

1.2.1 Servidor web Apache

El servidor HTTP Apache és un servidor web de codi obert per a plataformes Unix (BSD, GNU/Linux, etc.), Windows, Macintosh i altres sistemes operatius que implementa el protocol HTTP i que utilitza el concepte de lloc virtual.

L'Apache va començar com una sèrie de mòduls desenvolupats per a un servidor web en el qual treballaven al Centre Nacional de Supercomputació d'Aplicacions dels Estats Units (NCSA). Un cop va acabar aquest projecte, programadors de tot el món van trobar que hi havia la necessitat de tenir un repositori central en què es pogués mantenir el codi i els mòduls que s'havien desenvolupat per poder continuar treballant en el desenvolupament del servidor. Així va sorgir la Fundació de Programari Apache (Apache Software Foundation).

L'Apache va ser dissenyat des del principi de forma modular, de manera que el programari podia ser ampliat per altres desenvolupadors que escrivien petites parts del codi d'una manera fàcil. La clau d'aquest èxit és la creació d'una API (interfície de programació d'aplicacions) modular i una sèrie de fases ben definides que travessa cada petició que arriba al servidor. Aquestes fases van des de la inicialització del servidor (l'Apache quan llegeix els fitxers de configuració), fins a la traducció d'una adreça URL en un nom de fitxer de l'ordinador.

1.2.2 Instal·lació d'Apache

La forma general d'instal·lar paquets als sistemes basats en Debian és:

```
1 # sudo apt-get install {nom_del_paquet}
```

Per instal·lar paquets en distribucions basades en Debian GNU/Linux, s'han d'executar amb drets de *root* o amb un usuari pertanyent al grup *sudo*.

Per instal·lar el servidor web heu d'instal·lar el paquet **apache2**. Per defecte el servidor fa servir el directori **/var/www** per contenir les pàgines que servirà. Una vegada instal·lat podeu comprovar el seu funcionament modificant el fitxer **/var/www/index.html** i comprovant que es mostra correctament si obriu des del navegador web l'adreça **http://localhost** tal com es pot veure a la figura 1.4.

FIGURA 1.4. Comprovació del funcionament del servidor Apache



1.2.3 Configuració bàsica d'Apache

Es pot controlar l'arrencada i aturada del servidor Apache 2 amb un script que es troba dins de la carpeta **/etc/init.d**. La sintaxi d'aquest script és la següent:

```
1 /etc/init.d/apache2 ordre
```

en el qual **ordre** pot ser alguna de les següents opcions:

- start: per iniciar el servidor
- stop: per aturar el servidor
- restart: per reiniciar el servidor. Atura i engega el servidor.

Recordeu que heu de treballar amb permisos de superusuari per modificar l'execució del servidor, així que feu servir **sudo** o treballeu com a usuari *root*. Podeu comprovar com el servidor s'atura i engega entrant a la pàgina de *localhost*.

Un cop finalitzada la instal·lació d'Apache 2.0, ja en podem treure algunes conclusions. En primer lloc, podem afirmar que utilitza un disseny modular (basat en mòduls) que s'utilitza en moltes de les funcions bàsiques del servidor web. D'altra banda, tal com mostra el procés d'instal·lació, s'ha instal·lat per defecte el paquet **apache2-mpm-worker**. Aquest paquet proporciona una versió molt més àgil i ràpida del servidor. De fet, els mòduls de multiprocessament (MPM, multi-processing modules) són els responsables de connectar amb els ports de xarxa de la màquina, acceptar les peticions i gestionar les respostes corresponents. En el cas dels servidors web, el port de funcionament estàndard és el 80. Val a dir que dins del procés d'instal·lació per defecte del servidor web Apache també hi ha la instal·lació del mòdul de directori d'usuari (*userdir*), ja que cada usuari disposarà d'un espai en el seu espai de disc (*home*) per crear i desar les seves pàgines web. Amb tot, si bé hi ha un ampli ventall de paquets que s'han d'instal·lar amb Apache per augmentar-ne l'eficiència, cal no deixar de banda la descàrrega de la documentació del programari, ja que esdevindrà de gran utilitat per a l'administrador. El paquet a instal·lar és concretament: **apache2-doc**.

Considerant que, en el cas que haguem instal·lat la documentació, haurem de tornar a arrencar el servidor web, també podem accedir a la documentació mitjançant la pàgina principal predefinida en el servidor web Apache. Per tant, des de qualsevol navegador web ens podrem documentar sobre el funcionament del servidor web escrivint en la barra de navegació `http://localhost/manual/`.

1.2.4 Configuració de directoris personals

A partir d'ara, qualsevol fitxer que posem a `/var/www` s'ha de visualitzar quan escrivim en el navegador `http://IP/nom_del_fitxer`, en què IP és la IP de la màquina en la qual hi ha el servidor web instal·lat.

Podeu activar el directori personal dels usuaris de la següent manera:

1. Creeu una carpeta dins del directori personal de l'usuari anomenada `public_html`. Ho heu de fer amb el vostre usuari, no feu servir l'usuari *root* o *sudo*.

2. Creeu dins de la carpeta /mods-enabled d'Apache uns enllaços simbòlics cap als mòduls que voleu fer servir, tal com es pot veure a la figura 1.5.
3. Reinicieu el servidor.

En aquest moments el servidor us donarà el missatge que podeu veure a la figura 6 quan l'inicieu. Això és perquè el servidor no té un nom de domini totalment qualificat. És normal fins que no ho configureu.

FIGURA 1.5. Activació dels directoris dels usuaris

```
eduard@ASIXM9:/etc/apache2/mods-enabled$ sudo ln -s ../mods-available/userdir.conf userdir.conf
eduard@ASIXM9:/etc/apache2/mods-enabled$ sudo ln -s ../mods-available/userdir.load userdir.load
eduard@ASIXM9:/etc/apache2/mods-enabled$ sudo /etc/init.d/apache2 restart
Restarting web server: apache2apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1 for ServerName
... waiting apache2: Could not reliably determine the server's fully qualified domain name, using 127.0.1.1 for ServerName
```

A partir d'aquest moment l'usuari pot ficar el contingut web personal dins de la carpeta public_html que ha creat al seu directori personal. Aquesta serà accessible a través de l'adreça `http://localhost/~nom_usuari`. Per defecte mostrarà el fitxer index.html o un llistat dels fitxers del directori si no troba aquest fitxer.

1.2.5 Configuració de mòduls

Apache és un servidor modular. Això vol dir que en el cos del servidor està inclòs únicament les funcionalitats més bàsiques. Les funcionalitats extres estan disponibles mitjançant els mòduls que es poden carregar a l'Apache. Si el servidor està compilat per fer servir els mòduls dinàmicament, aquest es poden afegir en qualsevol moment. Si no és així, Apache s'ha de recompilar per afegir o treure els mòduls. Podeu instal·lar mòduls d'Apache amb l'ordre `apt-get`. Per exemple, per instal·lar el mòdul per autenticació MySQL podeu fer servir la següent ordre:

```
1 sudo apt-get install libapache2-mod-auth-mysql
```

Una vegada instal·lat, el mòdul estarà disponible a **/etc/apache2/mods-available**. Els mòduls que s'estan fent servir estan a **/etc/apache2/mods-enabled**.

Per activar un mòdul podeu fer un enllaç simbòlic al mòdul de **mods-available**.

```
1 sudo ln -s /etc/apache2/mods-available/auth_mysql.load /etc/apache2/mods-enabled/auth_mysql.load
```

o podeu fer servir l'ordre:

```
1 sudo a2enmod auth_mysql
```

Per desactivar el mòdul podeu fer servir:

```
1 sudo a2dismod auth_mysql
```


Podeu mirar els mòduls que està fent servir Apache mirant directament el contingut del directori de mòduls actius (**/etc/apache2/mods-enabled**) o amb l'ordre:

```
1 sudo apache2ctl -M | sort
```

1.2.6 Control d'accessos

Per autenticar als usuaris a Apache es pot fer de dues maneres Bàsica i Digest. A la Bàsica l'usuari introdueix en el navegador web el seu nom d'usuari i contrasenya i s'envien al servidor sense xifrar, també és la més senzilla de configurar. A la Digest l'usuari i contrasenya i s'envien al servidor xifrats.

Aquests mètodes controlen l'accés als recursos, però no xifren la comunicació client-servidor un cop s'ha validat l'accés.

Autenticació bàsica

El mòdul d'Apache que controla aquest mètode d'autenticació és **mod_auth_basic**.

- Està suportat per tots els navegadors web.
- L'usuari i la contrasenya no van xifrades del navegador web al servidor.

Per cada directori que es vulgui protegir a l'arxiu **/etc/apache2/sites-available/default**, o en el fitxer relatiu al *host* virtual corresponent, caldrà afegir un bloc `<Directory> ...</Directory>`:

```
1 <Directory "/var/www/privat">
2 AuthType Basic
3 AuthName "Directori privat"
4 AuthUserFile /etc/apache2/passwd/.htpasswd
5 Require valid-user
6 </Directory>
```

En el qual:

- **AuthName**: indica el nom del domini d'autenticació i és text que se li presentarà a l'usuari en el moment d'autenticar-se.
- **AuthType**: indica el mètode d'autenticació a usar.
- **AuthUserFile**: indica la ruta a l'arxiu de text que contindrà els noms d'usuari i contrasenyes usades en l'autenticació HTTP bàsica.
- **Require**: indica restriccions sobre els usuaris que tenen accés als recursos especificats. Pot ser *valid-user* que identifica qualsevol usuari inclòs a l'arxiu de contrasenyes *.htpasswd* o *user* `<llista d'usuaris>` que especifica la llista d'usuaris de *.htpasswd* que poden accedir.

1.3 Llenguatges script

Un llenguatge d'script (guions) és un llenguatge de programació basat en guions que són seguits línia per línia per un processador.

Un script és un conjunt d'instruccions que afecten a la pàgina web, si aquestes instruccions afecten al navegador es troben al costat del client i si afecten al servidor es troben a l'ordinador servidor. Si penseu en pàgines web que heu visitat recentment que canvien cada vegada que les visiteu o fins i tot que poden canviar mentre les esteu visitant és possible que utilitzin algun script.

Per exemple, quan busqueu una paraula al buscador Google, aquest utilitza scripts per suggerir paraules relacionades, per posar anuncis i per trobar el que esteu buscant. Quan compreu a una botiga virtual, aquesta utilitza scripts per mostrar els productes i per guardar el vostre carro de compra.

A continuació veureu les característiques principals dels dos tipus de llenguatges de guions.

1.3.1 Llenguatges script de client

Els llenguatges script de client s'executen al sistema en el qual es troba el navegador web (Firefox, Chrome, Opera, Safari, etc.) de l'usuari i poden formar part del codi HTML del document o trobar-se en arxius adjunts. Els llenguatges script de client donen interactivitat a les pàgines HTML, el codi és lleuger i al arribar al navegador és interpretat (és a dir, els scripts s'executen sense una compilació preliminar) pel navegador.

El procés d'un navegador al sistema del client és:

1. L'usuari demana una pàgina web a un servidor des del seu navegador.
2. El servidor cerca la plana i, si la troba, la retorna al navegador de l'usuari.
3. El navegador mostra la pàgina i al mateix temps executa els scripts de client per crear la sortida HTML.

Els llenguatges d'script de client modifiquen la pàgina web un cop arriben al navegador. L'avantatge principal dels scripts de client és que al no executar-se al servidor no consumeixen CPU del servidor ni ample de banda. S'utilitzen per a tasques que estalvien feina al servidor, com ara validar els camps d'un formulari, o per millorar l'aparença i la interacció amb l'usuari.

El llenguatge de referència per programar codi en el cantó del client és **JavaScript**. No es necessita cap llicència per utilitzar-lo. Al executar-se a la màquina del client si aquesta és lenta els scripts es poden executar lentament, o fins i tot no s'arribaran

a executar si el navegador no entén el llenguatge d'script. A les preferències dels navegadors moderns es pot activar o desactivar l'execució de Javascript. Com que el codi és a l'ordinador del client l'usuari el pot veure per copiar-lo o manipular-lo.

Amb la tecnologia **AJAX** i el Javascript, la millora de funcionalitat de les pàgines web al navegador fa que s'assemblin cada cop més a aplicacions normals de l'ordinador.

AJAX

Són les sigles d'*Asynchronous Javascript And Xml*, (JavaScript asíncron i XML), un conjunt de tecnologies que permeten actualitzar parts d'una pàgina web sense haver de tornar-la a carregar sencera.

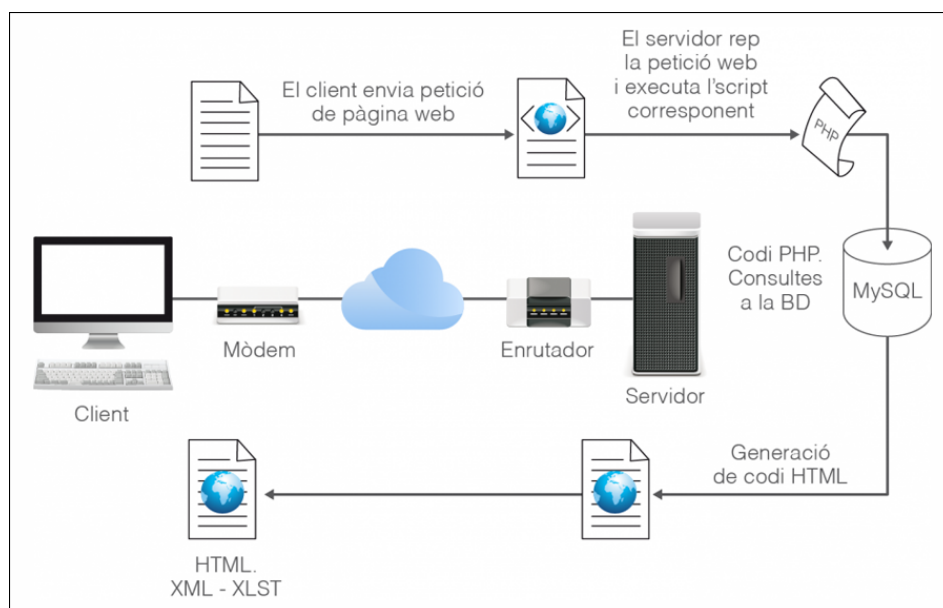
1.3.2 Llenguatges script de servidor

Les aplicacions web desenvolupades amb llenguatges script de servidor s'executen en el servidor web i per tant l'usuari no té accés al codi font del servidor. El servidor conté aquests scripts que són interpretats cada vegada que es rep una petició d'una pàgina web. El procés és el següent:

1. L'usuari a través del seu navegador demana una pàgina al servidor.
2. L'script de servidor és interpretat muntant el contingut a retornar (la pàgina web).
3. Quan l'script acaba, el contingut muntat es retorna al client.

A l'esquema de la figura 1.6 s'il·lustra una màquina client que envia una petició de pàgina web a un servidor que executa un script de servidor que requereix consultar una base de dades. Com a resposta, el client rep una pàgina web HTML.

FIGURA 1.6. Funcionament dels llenguatges script de servidor



Entre les característiques més significatives dels llenguatges script de servidor es troben les de permetre als usuaris tenir comptes individuals i la de comunicar-se amb bases de dades (MySQL, PostgreSQL, SQL Server, Oracle, etc). Permeten

un nivell de personalització, privacitat i recuperació d'informació molt important. Les pàgines web de comerç electrònic i les pàgines web socials fan un ús intensiu dels scripts de servidor.

Com a exemples dels principals llenguatges script de servidor tenim el PHP (pàgines amb extensió php) i l'ASP/ASP.net (pàgines amb extensió asp/aspx). Altres llenguatges són Perl (pàgines amb extensió pl), Java Server Pages (pàgines amb extensió jsp), ColdFusion (pàgines amb extensió cfm) i Python (pàgines amb extensió py).

1.3.3 Instal·lació de PHP5

Hi ha una sèrie de paquets necessaris per treballar amb PHP i Apache. Els podeu instal·lar amb `apt-get` (php5, php5-common, libapache2-mod-php5 php5-cli).

Una vegada instal·lats heu de reiniciar el servidor Apache perquè els canvis siguin efectius.

De manera opcional podeu instal·lar del PHP5 els paquets referenciats aquí per donar suport a diferents utilitats. Si sols necessiteu una instal·lació bàsica del PHP, no fa falta instal·lar els paquets següents, encara que són recomanables.

- php5-gd: biblioteca per a l'ús d'imatges.
- libphp-adodb: biblioteca adodb per a PHP.
- Smarty: sistema Templates PHP.
- php-pear: biblioteques Pear per a PHP.
- php5-json: suport per a objectes JSON.
- php5-xsl: suport per a XSLT.
- php5-odbc: per a connexions odbc.

1.4 MySQL

El MySQL és un sistema de gestió de base de dades relacional. Es calcula que té més de 6 milions d'instal·lacions. El programa s'executa com un servidor multiusuari i proporciona accés a una sèrie de bases de dades.

El projecte de codi font està disponible en els termes de la llicència pública general GNU, tot i que el MySQL és propietat d'una empresa amb ànim de lucre, que també el patrocina. El MySQL era propietat de la companyia sueca MySQL AB, una empresa que Sun Microsystems va comprar el 2008. L'abril de 2009, Oracle

va comprar Sun i indirectament MySQL AB, un dels principals competidors en alguns dels seus productes.

El MySQL és utilitzat per una gran quantitat de projectes que requereixen una base de dades amb funcions completes, sistema de gestió, administració i és usat, entre d'altres, per WordPress, phpBB, Google i Facebook.

Moltes vegades es fa referència a la combinació de Linux, el servidor web Apache, el contenidor web de PHP i MySQL amb l'acrònim *LAMP*.

1.4.1 Instal·lació de MySQL a Debian

La forma més senzilla d'instal·lar MySQL a Debian és mitjançant `apt-get`. Heu d'instal·lar els paquets *mysql-client* i *mysql-server*. Us demanarà quina voleu que sigui la contrasenya de *root* de MySQL. MySQL té els seus propis usuaris, que són independents dels usuaris del sistema Linux. Per tant és important canviar la contrasenya de *root* i apuntar-la.

Anoteu la contrasenya de *root* en un lloc segur, ja que la necessitareu més endavant.

Si més endavant voleu tornar a canviar la contrasenya ho podeu fer amb l'ordre:

```
1 $ sudo mysqladmin -u root -h localhost password 'clau'
```

Podeu canviar *localhost* pel nom de la màquina. Un cop fet això, ja podeu entrar al servidor MySQL amb l'ordre:

```
1 $ mysql -u root -p
```

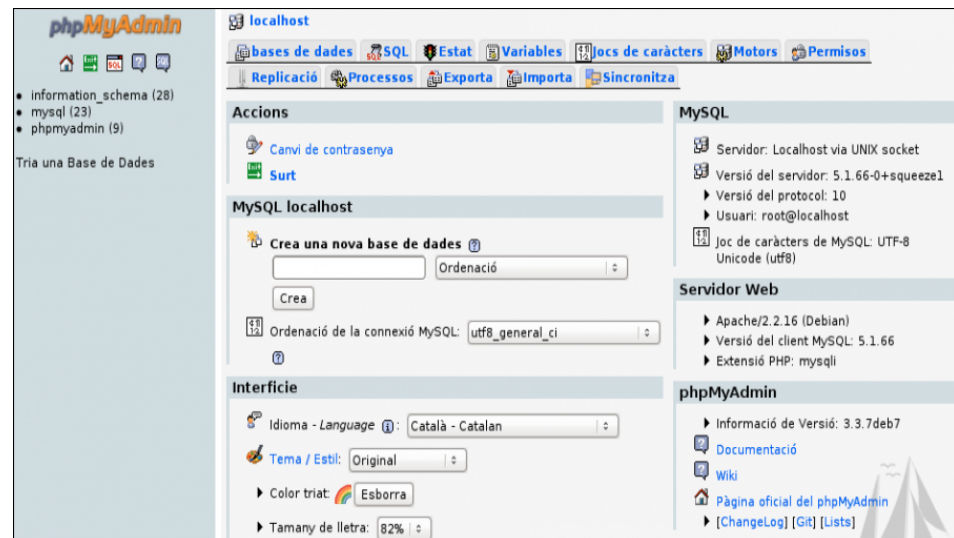
Per aturar o engegar el servidor de MySQL teniu l'script que el controla a */etc/init.d/mysql* que accepta les típiques opcions (*start*, *stop*, *restart*).

Assegureu-vos que teniu instal·lat el paquet *php5-mysql* ja que el necessitareu per treballar amb MySQL i PHP. Ho podeu comprovar amb l'ordre:

```
1 $ dpkg -s php5-mysql
```

1.5 phpMyAdmin

El phpMyAdmin és un programa de distribució lliure en PHP, creat per una comunitat sense ànim de lucre. És una eina molt completa que permet accedir a totes les funcions típiques de la base de dades MySQL per mitjà d'una interfície web molt intuïtiva, tal com es pot veure a la figura 1.7.

FIGURA 1.7. Pantalla principal de l'aplicació phpMyAdmin

L'aplicació tan sols és un conjunt d'arxius escrits en PHP que es copien en un directori del servidor web, de manera que, quan s'accedeix a aquests arxius, trobem una eina que ens permet crear taules, inserir dades a les taules existents, navegar pels registres de les taules, editar i esborrar, esborrar taules, etc. Fins i tot executar sentències SQL i fer una còpia de seguretat de la base de dades.

1.5.1 Instal·lació del phpMyAdmin

La pàgina d'inici del projecte és www.phpmyadmin.net. Des d'aquí podeu baixar els fitxers de la darrera versió de l'aplicació, que després heu de col·locar en el nostre servidor web.

Per a distribucions GNU/Linux basades en Debian, també es pot fer la instal·lació directament des del repositori:

```
1 $ sudo apt-get install phpmyadmin
```

Durant la instal·lació us farà una sèrie de preguntes:

- Pel que fa al servidor web que farà servir heu de contestar: `apache2`.
- Quan us pregunta si voleu configurar la base de dades phpMyAdmin amb `dbconfig-common`, heu de contestar que `sí`.
- Quan us pregunta la contrasenya de `root` de MySQL (perquè phpMyAdmin vol crear les seves taules a la base de dades MySQL)
- Us demanarà la clau d'accés a la base de dades MySQL de l'usuari que s'ha creat pel phpMyAdmin. Podeu escriure una clau vosaltres o, si la deixeu en blanc, crearà una clau aleatòria.

Abans de poder accedir al panell de control de phpMyAdmin s'ha de modificar la configuració d'Apache.

Trobeu on és el fitxer `apache.conf`. Generalment l'hauríeu de tenir a `/etc/phpmyadmin/apache.conf`

En qualsevol cas el podeu trobar amb les ordres:

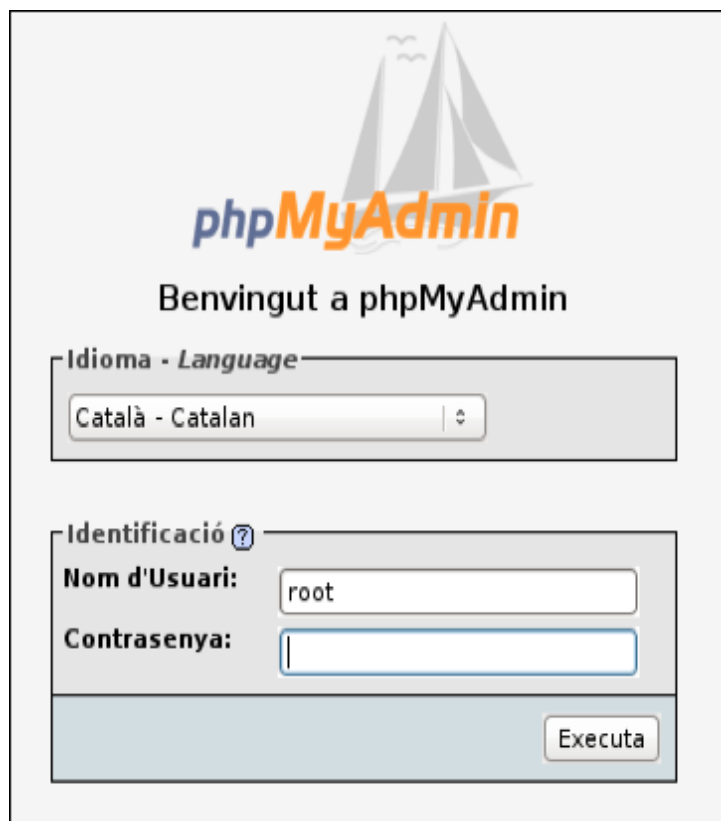
```
1 $ updatedb
2 $ locate apache.conf
```

Editeu la configuració d'Apache i incloeu-hi el fitxer `apache.conf` de phpMyAdmin. Per això obriu (amb drets de *root*) el fitxer `/etc/apache2/apache2.conf` i afegiu al final del fitxer la següent línia:

```
1 Include /etc/phpmyadmin/apache.conf
```

Ara ja podeu reinicar el servidor Apache i anar a l'adreça <http://localhost/phpmyadmin> per veure si funciona. Hauríeu de trobar una pantalla com la que es pot veure a la figura 1.8.

FIGURA 1.8. Pantalla d'inici de phpMyAdmin



1.6 Utilitats de prova i instal·lació integrada

La instal·lació i posterior configuració d'un servidor web Apache, d'un servidor de bases de dades MySQL i del llenguatge de programació PHP és una tasca complexa que només poden emprendre usuaris amb bons coneixements informàtics.

La primera lletra fa referència al sistema operatiu: LAMP (Linux), WAMP (Windows), o MAMP (Mac OS).

Els anomenats paquets LAMP, WAMP o MAMP simplifiquen la tasca d'instal·lar i configurar automàticament Apache, PHP i MySQL en Linux, Windows o Mac OS.

Aquests paquets proporcionen:

- Servidor Web Apache
- Base de dades MySQL
- Llenguatge de programació PHP
- Accessos per l'arrencada i la parada dels serveis
- Facilitats per a la configuració dels serveis

El paquet XAMPP és un paquet de programari lliure que té versions per Linux, Windows, Solaris i Mac OS X, a més d'oferir els serveis bàsics d'un paquet integrat (és a dir, Servidor web Apache + Servidor de bases de dades MySQL + Llenguatge PHP). Així doncs, ofereix:

- ProFTPD com a servidor d'arxius per FTP
- Llenguatge Perl
- Servidor de dades SQLite
- SSL per pàgines segures HTTPS (OpenSSL)
- Estadístiques d'accés (Webalizer)

Podeu trobar la darrera versió de XAMPP a www.apachefriends.org/en/xampp.html

1.6.1 Utilitats de prova

Per comprovar si funciona la instal·lació d'Apache2 i PHP:

Creeu un document a l'arrel en **/var/www**, anomenat **info.php** i empleneu-lo amb la informació següent:

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

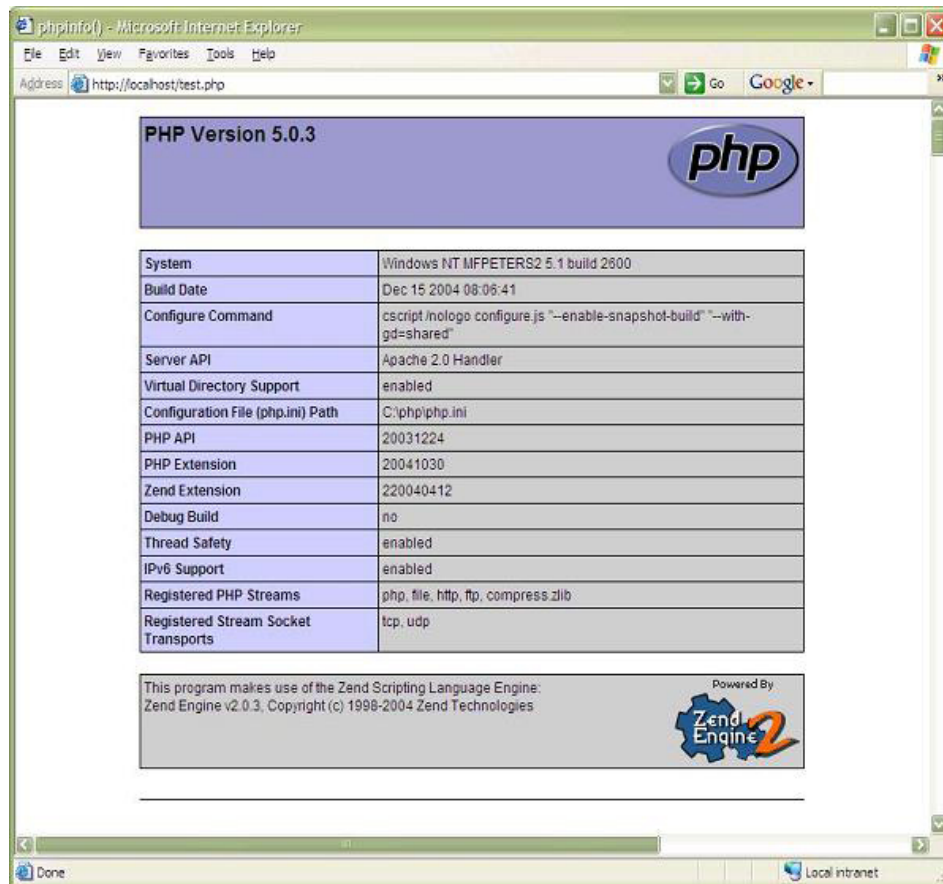
A continuació, deseu l'arxiu.

Proveu el servidor amb el navegador i la direcció del tipus següent:

```
1 http://localhost/info.php
```


Si tot ha anat bé, us mostrarà una pantalla amb informació del PHP (veure figura 1.9):

FIGURA 1.9. Informació de configuració de PHP



Per comprovar, un cop acabada la instal·lació de MySQL, que podeu accedir a l'entorn, entreu com a usuari *root* i executeu l'ordre *show Databases*. Veureu les dues bases de dades que instal·la MySQL i executeu *quit* per sortir de l'entorn.

A un terminal escriureu la comanda següent:

```

1  elatorre@debian:~$ mysql -h localhost -u root -p
2  Enter password:
3  Welcome to the MySQL monitor. Commands end with ; or \g.
4  Your MySQL connection id is 41
5  Server version: 5.1.49-3 (Debian)
6
7  Copyright (c) 2000, 2010, Oracle and/or its affiliates. All rights reserved.
8  This software comes with ABSOLUTELY NO WARRANTY. This is free software,
9  and you are welcome to modify and redistribute it under the GPL v2 license
10
11  Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
12
13  mysql> show databases;
14  +-----+
15  | Database |
16  +-----+
17  | information_schema |
18  | mysql |
19  +-----+
20  2 rows in set (0.00 sec)
21  mysql> quit;
```

```
22 Bye  
23 elatorre@debian:~$
```

1.7 Documentació

Durant el procés d'instal·lació i configuració d'un servidor d'aplicacions web, cal prendre nota de tots els passos que s'han seguit, primer en el moment de la instal·lació i després en el moment de la configuració, de manera que tingueu una referència completa de tot el que heu fet. Conèixer els detalls de per què s'ha posat un valor a una variable o a un fitxer de configuració pot ser molt útil en el futur, ja que és probable que passat un temps us n'oblíeu.

Crear la documentació és una part important del procés i pot ser molt útil quan apareixen dificultats, incidències o en manteniments, i es volen saber ràpidament els valors dels paràmetres configurats sense haver d'anar a mirar directament els arxius de configuració al servidor.

La documentació també és importat per a la transmissió de coneixement. Podeu passar la documentació a altres persones que poden aprendre com està muntat el servidor sense haver d'explicar res per la vostra part.

Quan es vol replicar en una altra màquina la instal·lació i configuració d'un servidor web, la documentació és una gran ajuda per saber com s'ha fet i com està muntat el servidor que ja està en funcionament.

Una bona manera de començar a fer la documentació és consultar l'índex que heu seguit per fer la instal·lació de l'aplicació. Es poden analitzar els punts de l'índex que s'han anat fent durant el procés d'instal·lació, fer-ne un resum i documentar especialment les parts en les quals es difereix de la guia que s'ha fet servir per fer la instal·lació.

1.7.1 Eines d'ajuda a la creació

Una eina útil que podeu fer servir per crear la documentació és un wiki.

Un wiki (del hawaià *wikiwiki*, que vol dir *ràpid*) és un lloc web col·laboratiu que pot ser editat des del navegador pels usuaris. Els usuaris d'un wiki poden, d'aquesta manera, crear, modificar, enllaçar i esborrar el contingut d'una pàgina web, de forma interactiva, fàcil i ràpida. Un dels wikis més coneguts és la Viquipèdia (*Wikipedia*) (veure figura 1.10).

FIGURA 1.10. Pàgina web de la Viquipèdia



Els projectes wiki tenen normalment historials amb totes les modificacions que s'han fet dels seus continguts. D'aquesta manera se'n poden veure totes les versions i recuperar la informació eliminada o desfer les edicions incorrectes, ja que els canvis s'apliquen normalment a l'instant, sense que cap mena d'usuari de confiança o administrador els hagi revisat i confirmat abans.

Fins ara, l'aplicació dels sistemes wiki deuen la seva fama sobretot a la creació d'enciclopèdies col·lectives com la Viquipèdia. Però les característiques dels wikis els converteixen en una eina efectiva per a l'escriptura col·laborativa, i cada vegada són més usades en empreses com a webs i intranets econòmiques i eficaces per a la gestió del coneixement.

Hi ha molts programaris de wiki diferents (podeu consultar l'enllaç http://en.wikipedia.org/wiki/Comparison_of_wiki_software), i que fins i tot podeu instal·lar al mateix servidor web. Entre els més recomanables trobem els programaris DokuWiki i MediaWiki, que fan servir una sintaxi molt similar.

El **DokuWiki** (veure logo figura 1.11) està llicenciat sota GPL 2 i escrit en el llenguatge de programació PHP. Funciona amb arxius de text pla i per tant no necessita cap base de dades.

FIGURA 1.11. Logo DokuWiki



El **MediaWiki** (veure logo figura 1.12) és el programari de wiki amb el qual s'ha fet la Viquipèdia, està llicenciat sota llicència GPL, està escrit en el llenguatge de programació PHP i utilitza MySQL com a gestor de base de dades.

FIGURA 1.12. Logo MediaWiki



2. Programació web de servidor

La programació web del servidor es pot realitzar fent servir diferents eines i tecnologies. PHP és un dels llenguatges més populars de programació web del servidor, generalment integrat dins de codi web codificat en HTML. En les darreres versions de PHP es pot fer servir el paradigma d'orientació a objectes.

2.1 El llenguatge PHP

El **PHP** és un llenguatge de programació obert que s'integra directament en pàgines HTML. El seu ús ha esdevingut gairebé imprescindible en les pàgines que han d'utilitzar bases de dades o formularis.

Rasmus Lerdorf és el creador del llenguatge PHP. El 1995 presenta la primera edició d'aquest llenguatge amb el nom de *Personal Home Page tools* amb llicència pública GNU. Per crear el PHP, Rasmus utilitza el codi de programació C.

Tot i que el PHP és un llenguatge ideat i creat per explotar tots els avantatges d'Internet, la seva fisonomia permet fer altres tasques potser no relacionades directament amb la xarxa de xarxes. Es podria afirmar que el PHP permet al programador fer tot tipus de programes sense dependre d'altres llenguatges.

Per utilitzar el PHP cal disposar d'un **navegador** web, un **servidor** web que suporti el PHP i instal·lar un **motor** PHP a la màquina en què es desenvoluparà el codi.

El PHP és compatible amb tots els navegadors web actuals (Firefox, Chrome, Safari, Internet Explorer i d'altres). Generalment, tots els proveïdors de serveis web suporten el PHP, però cal assegurar-se'n llegint les característiques del producte contractat.

2.1.1 Funcionament del llenguatge PHP

El codi PHP que escriviu, l'haureu d'incrustar dins de codi HTML, concretament dins les etiquetes `<body>` del document. Mitjançant unes marques l'HTML no interpretarà el codi PHP, el qual, i mitjançant les mateixes marques, s'executarà en el servidor.

Segons els motors principals de cerca a Internet, el PHP és el sisè llenguatge més popular, després del C, Java, Objective C, C++ i C#.

HTML

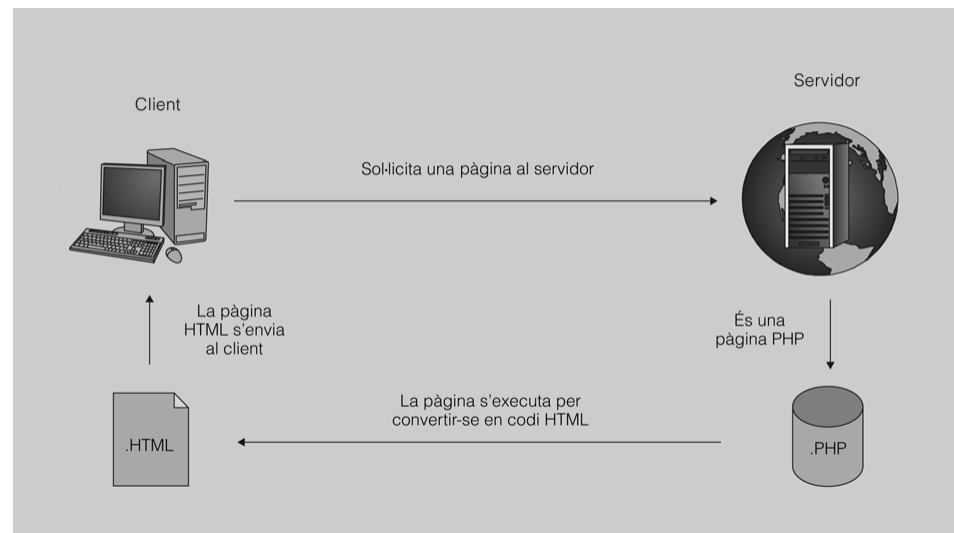
Hypertext markup language. És el llenguatge de marques que més es fa servir per construir pàgines web. L'extensió dels arxius HTML pot ser **.htm** o **.html**.

El codi següent és un exemple que mostra on es pot col·locar el codi PHP dins del codi HTML:

```
1 <html>
2 <head>
3   <title>Exemple amb PHP</title>
4 </head>
5 <body>
6 <?php
7   //Aquí és on incloureu el vostre codi en PHP
8   ?>
9 </body>
10</html>
```

El codi PHP s'executa en un servidor i mostra el resultat en un client. Visualitzareu els resultats amb un navegador web. En la figura 2.1 podeu veure el funcionament del llenguatge PHP i com interactuen el servidor i la màquina d'usuari.

FIGURA 2.1. Funcionament del llenguatge PHP



2.1.2 Configuració del llenguatge PHP

Quan arrenca el PHP es fa la lectura del fitxer *php.ini*. Aquest fitxer conté la configuració del PHP i es crea durant la instal·lació del PHP. És possible que necessiteu fer canvis en el fitxer de configuració, preneu la precaució de fer-ne una còpia de seguretat abans de fer aquestes modificacions.

Una inspecció visual del fitxer *php.ini* us servirà per comprendre'n el funcionament. Depenent del valor de les variables que conté aquest fitxer, el comportament del PHP variarà en la vostra màquina. Podeu comprovar que cada variable que forma part del fitxer està àmpliament comentada, amb comentaris que es marquen amb un punt i coma.

2.1.3 Eines de programació per a PHP

El programador de codi PHP haurà de fer un exercici previ per escollir l'entorn de programació que li sigui més còmode. A causa del bon acolliment d'aquest llenguatge de programació són molts els programes de desenvolupament de codi que han anat sorgint en els últims anys.

Com a desenvolupadors haureu de crear l'ambient de treball més adequat si voleu crear i produir codi de la millor manera possible. Escollir correctament l'entorn de desenvolupament integrat us serà de gran ajuda.

Per decidir l'eina de treball adequada haureu de tenir en compte diversos factors, com ara la facilitat d'ús, la capacitat d'integració, la possibilitat de treballar alhora amb altres llenguatges, el suport de l'eina, el desenvolupament del programari de l'eina, la possibilitat real de fer servir utilitats com repositoris i, sobretot, que vosaltres us trobeu còmodes treballant amb aquest entorn.

Repositori

Espai en què s'emmagatzema informació d'una manera centralitzada. Normalment, relacionem repositoris amb control de versions, el qual crea un registre dels canvis efectuats en fitxers, permet recuperar versions antigues i possibilita l'accés a les últimes modificacions. La importància de l'ús de repositoris i control de versions augmenta a mesura que el projecte de programació es fa més complex.

Abans de decidir quin entorn de programació fareu servir proveu-ne uns quants. Els programes que més es fan servir són el Komodo IDE, NetBeans, Eclipse, Adobe Dreamweaver o Aptana.



Logo de NetBeans

Què és un IDE?

Un entorn de desenvolupament integrat (*Integrated Development Environment*) és un programa que aglutina un conjunt d'eines de gran utilitat per al programador.

2.1.4 Sintaxi del llenguatge PHP

Si voleu que s'interpreti el vostre codi és imprescindible que encabiu el codi escrit en PHP entre dos delimitadors. Aquestes marques són `<?php`, que la fareu servir per indicar l'inici de codi en llenguatge PHP, i `?>`, que utilitzareu per marcar el final del vostre codi PHP.

Aquestes marques són les més genèriques i les més recomanades, però no les úniques que es poden emprar. Així, per marcar els blocs de codi PHP, també podreu fer servir `<?` per indicar l'inici del codi, i `?>` per marcar el final; ara bé, per poder fer servir aquestes marques, tingueu en compte que probablement hàgiu de modificar la configuració del PHP mitjançant el fitxer `php.ini`.

Els documents XHTML i XML necessiten la forma `<?php ?>` per interpretar la codificació PHP. Gran part de les pàgines web dissenyades avui dia fan servir aquests formats; per tant, si voleu que el vostre codi PHP s'interpreti correctament, feu servir aquestes marques.

L'XHTML i l'XML aconsegueixen nivells alts de compatibilitat entre diferents sistemes d'una manera molt senzilla i segura.

Instruccions

En PHP indicareu el final d'una instrucció amb punt i coma. Cada vegada que es detecti un punt i coma s'interpretarà el codi inserit fins a aquell punt.

El codi següent és un exemple d'instrucció. L'ordre `echo` mostrarà en el vostre navegador el text escrit entre cometes dobles.

Podeu fer servir cometes dobles o simples per indicar la sentència, més endavant veureu les conseqüències que se'n deriven.

```
1 <?php
2 echo "Aquí teniu una instrucció.";
3 ?>
```

Les seqüències de control de codi PHP són les úniques que no finalitzen amb punt i coma.

A continuació, teniu la manera més correcta d'escriure dues instruccions. En aquest cas apareix `
`. Aquesta és una etiqueta bàsica de l'HTML que provoca un salt de línia.

```
1 <?php
2 echo "Aquí teniu una instrucció.";
3 echo "<br>i aquí una altra instrucció.";
4 ?>
```

Comentaris

Amb l'objectiu de crear un codi més entenedor utilitzareu comentaris. Els comentaris en PHP els podreu incloure mitjançant:

- Dues barres (`//`) o un coixinet (`#`) si voleu comentar una línia
- Una barra i un asterisc (`/*`) per indicar l'inici de comentari, i un asterisc i una barra per indicar el final de comentari si voleu comentar més d'una línia(`/*`).

```
1 <?php
2 //Si voleu comentar una línia ho podeu fer d'aquesta manera...
3 #...o bé d'aquesta altra.
4 /* Però si necessiteu comentar
5 més línies aquesta és la millor
6 manera de fer-ho! */
7 ?>
```

No poseu comentaris aniuats, us causaran problemes.

És recomanable incloure comentaris en el vostre codi. És important per ajudar-vos en la comprensió del codi i també per recordar els motius del disseny del codi, però també és important per fer-lo més entenedor a la resta de programadors.

2.1.5 Tipus de dades del llenguatge PHP

El PHP és un llenguatge de programació que no obliga a declarar prèviament el nom de les variables que s'utilitzaran ni el seu tipus (és diu que és un llenguatge

sense tipus o no tipat). Ara bé, tot i que existeixen els tipus de dades en PHP aquests no es comproven. Una variable pot emmagatzemar dades del tipus enter, decimal, lògic, cadenes de text, col·leccions o objectes. Així si emmagatzemeu a una variable una cadena de text i més endavant li assigneu una variable entera, no donarà error.

Quan treballem amb objectes aneu amb compte, ja que les diferents versions del PHP no es comporten igual amb aquest tipus de dades.

Programació orientada a objectes i PHP

Des de la versió 5 de PHP, aquest es pot fer servir com llenguatge orientat a objectes, tot i que el programador no està obligat a fer servir aquesta metodologia de programació.

2.1.6 Operadors

Els operadors us permetran fer accions dins el codi. Podreu canviar i assignar valors, canviar l'adreça del codi, condicionar l'execució de blocs de codi i, en definitiva, dotar de la complexitat necessària els vostres programes per aconseguir un objectiu.

Els operadors es poden agrupar en tres blocs segons el nombre de valors sobre els quals s'actua.

1. L'**operador unari** opera sobre un valor i el que pot fer en aquest valor és negar-lo, incrementar-lo o decrementar-lo, entre altres coses.
2. L'**operador binari** opera sobre dos valors i permet sumar-los, restar-los o comparar-los.
3. L'**operador ternari** permet escollir entre dues expressions en funció d'una altra.

Teniu en compte la precedència dels operadors.

En la taula 2.1 apareixen els **operadors aritmètics** de què disposeu. Teniu en compte que la divisió retornarà un valor en coma flotant si el resultat de la divisió no és exacte.

TAULA 2.1. Operadors aritmètics

Operador	Nom	Acció	Tipus d'operador
-\$valor	Negació	Oposat de \$valor	Operadors d'aritmètica
\$valor + \$valorB	Addició	Suma de \$valor i \$valorB	Operadors d'aritmètica
\$valor - \$valorB	Substracció	Diferència entre \$valor i \$valorB	Operadors d'aritmètica
\$valor * \$valorB	Multiplicació	Producte de \$valor i \$valorB	Operadors d'aritmètica
\$valor / \$valorB	Divisió	Quocient de \$valor i \$valorB	Operadors d'aritmètica
\$valor % \$valorB	Mòdul	Reste de \$valor dividit per \$valorB	Operadors d'aritmètica

En la taula 2.2 es presenten els operadors d'assignació. L'operador bàsic d'assignació és l'igual. Amb l'operador igual la variable de l'esquerra rep el valor de l'expressió de la dreta.

TAULA 2.2. Operadors d'assignació

Operador	Nom	Acció	Tipus d'operador
\$valor = \$valorB	Assignació	Assignar el contingut de \$valorB a \$valor	Operadors d'assignació
\$valor += \$valorB	Assignació	Sumar el contingut de \$valorB i \$valor i emmagatzemar-lo a \$valor	Operadors d'assignació
\$valor.= "text"	Assignació	Concatenar la cadena de la dreta a la de l'esquerra	Operadors d'assignació

Per activar o desactivar bits individuals d'un enter fareu servir els **operadors de bit**. En la taula 2.3 es mostren els operadors de bit de què disposeu.

TAULA 2.3. Operadors de bit

Operador	Nom	Acció	Tipus d'operador
\$valor & \$valorB	I	Els bits que són actius tant en \$valor com en \$valorB són activats.	Operadors de bit
\$valor \$valorB	O	Els bits que són actius en \$valor o en \$valorB són activats.	Operadors de bit
\$valor ^ \$valorB	XOR	Els bits que són actius en \$valor o \$valorB, però no en tots dos, són activats.	Operadors de bit
~ \$valor	No	Es canvia l'estat dels bits.	Operadors de bit
\$valor << \$valorB	Desplaçament a l'esquerra	Desplaça els bits de \$valor, \$valorB posicions a l'esquerra.	Operadors de bit
\$valor >> \$valorB	Desplaçament a la dreta	Desplaça els bits de \$valor, \$valorB posicions a la dreta.	Operadors de bit

Per comparar valors el llenguatge PHP disposa d'**operadors de comparació**. En la taula 2.4 teniu els casos possibles que podeu utilitzar.

TAULA 2.4. Operadors de comparació

Operador	Nom	Acció	Tipus d'operador
\$valor == \$valorB	Igual	TRUE si \$valor és igual a \$valorB.	Operadors de comparació
\$ valor === \$ valorB	Idèntic	TRUE si \$valor és igual a \$valorB i són del mateix tipus.	Operadors de comparació
\$ valor != \$ valorB	Diferent	TRUE si \$valor no és igual a \$valorB.	Operadors de comparació
\$ valor <> \$ valorB	Diferent	TRUE si \$valor no és igual a \$valorB.	Operadors de comparació
.			

TAULA 2.4 (continuació)

Operador	Nom	Acció	Tipus d'operador
\$ valor!=\$ valorB	No idèntic	TRUE si \$valor no és igual a \$valorB o no són del mateix tipus.	Operadors de comparació
\$ valor < \$ valorB	Més petit que	TRUE si \$ valor és més petit que \$ valorB.	Operadors de comparació
\$ valor > \$ valorB	Més gran que	TRUE si \$ valor és més gran que \$ valorB.	Operadors de comparació
\$ valor <= \$ valorB	Més petit o igual que	TRUE si \$ valor és més petit o igual que \$ valorB.	Operadors de comparació
\$ valor >= \$ valorB	Més gran o igual que	TRUE si \$ valor és més gran o igual que \$ valorB.	Operadors de comparació

En la taula 2.5 teniu els **operadors d'increment i decrement**, operadors que resulten de gran utilitat i simplifiquen força el codi.

TAULA 2.5. Operadors d'increment/decrement

Operador	Nom	Acció	Tipus d'operador
++\$valor	Preincrement	Incrementa \$valor una unitat, i després retorna \$valor.	Operadors d'increment/decrement
\$valor ++	Postincrement	Retorna \$valor i incrementa \$valor una unitat.	Operadors d'increment/decrement
-\$valor	Predecrement	Decrementa \$valor una unitat i retorna \$valor.	Operadors d'increment/decrement
\$valor--	Postdecrement	Retorna \$valor i decrementa \$valor una unitat.	Operadors d'increment/decrement

El PHP us proporciona **operadors de lògica**. En la taula 2.6 teniu les possibilitats de què disposeu. Els operadors *I* i *O* estan repetits perquè poden operar amb precedències diferents.

TAULA 2.6. Operadors de lògica

Operador	Nom	Acció	Tipus d'operador
\$valor and \$valorB	I	TRUE si \$valor i \$valorB són TRUE.	Operadors de lògica
\$valor or \$valorB	O	TRUE si \$valor o \$valorB són TRUE.	Operadors de lògica
\$valor xor \$valorB	XOR	TRUE si \$valor o \$valorB són TRUE, però no tots dos alhora.	Operadors de lògica
! \$valor	No	Inverteix el contingut de \$valor	Operadors de lògica
\$valor && \$valorB	I	TRUE si \$valor i \$valorB són TRUE.	Operadors de lògica
\$valor \$valorB	O	TRUE si \$valor o \$valorB són TRUE.	Operadors de lògica

Les matrius poden rebre l'acció d'operadors especials. En la taula 2.7 teniu els **operadors de matrius** que podeu utilitzar.

TAULA 2.7. Operadors de matrius

Operador	Nom	Acció	Tipus d'operador
\$ valor + \$ valorB	Unió	Unió de \$ valor i \$ valorB.	Operadors de matrius
\$ valor == \$ valorB	Igualtat	TRUE si \$valor i \$valorB tenen les mateixes parelles clau/valor.	Operadors de matrius
\$ valor === \$ valorB	Identitat	TRUE si \$valor i \$valorB tenen les mateixes parelles clau/valor en el mateix ordre i dels mateixos tipus.	Operadors de matrius
\$ valor != \$ valorB	No-identitat	TRUE si \$valor no és igual a \$valorB.	Operadors de matrius
\$ valor <> \$ valorB	No-identitat	TRUE si \$valor no és igual a \$valorB.	Operadors de matrius
\$ valor !== \$ valorB	No-identitat	TRUE si \$valor no és idèntic a \$valorB.	Operadors de matrius

2.1.7 Cadenes i matrius en PHP

Una cadena de text és una successió de caràcters. El tipus al qual pertanyen les cadenes és l'*string*, el qual no té inicialment suport per a Unicode, problema que se solucionarà amb funcions predefinides del mateix llenguatge.

En PHP les cadenes tenen una longitud màxima marcada per la memòria de la màquina en què s'executi el codi. Podreu especificar les vostres cadenes de quatre maneres diferents:

1. Amb cometes simples. Es considera que aquesta és la manera més senzilla d'indicar una cadena, però haureu de tenir en compte que no s'interpretaran les variables i les seqüències de sortida. Així, en l'exemple següent, no es mostrarà el valor de la variable *\$nom* ni tampoc es farà el salt de línia amb `\n`. Fixeu-vos, però, que sí que s'interpreta el salt de línia marcat amb l'etiqueta d'HTML `
`.

```

1 <?php
2 $nom='Abeeku';
3 echo 'Hola, company!<br>
4 Em dic $nom.\n
5 Què et sembla aquest crèdit?';
6 ?>
```

Cometes dobles

Vigileu amb les cometes dobles ja que, de vegades, quan copiem el codi i l'enganxem a l'editor es copia un caràcter de cometes diferent. Per exemple, si copieu un codi des del Microsoft Office, tindreu un caràcter de cometes diferent que no serà interpretat pel PHP.

2. Amb cometes dobles. Fent servir cometes dobles per marcar cadenes podreu mostrar el valor de les variables. El text inclòs entre cometes s'interpretarà amb l'HTML, per tant, seran les etiquetes d'HTML les que es podran interpretar. Escrivim un text que contingui una variable i una seqüència d'escapament en l'exemple següent. Comprovareu que es mostrarà el contingut de la variable.

```
1 <?php
2 $nom='Abeeku';
3 echo "Hola, company!<br>
4 Em dic $nom.\n
5 Què et sembla aquest crèdit?";
6 ?>
```

3. Utilitzant sintaxi *heredoc*. Quan necessiteu escriure un text molt llarg és recomanable que feu servir aquesta forma. Utilitzant aquesta sintaxi millorareu la velocitat del codi i facilitareu la lectura. En l'exemple següent podeu veure com s'utilitza la sintaxi *heredoc*. La marca que es fa servir per identificar el text és <<< seguida del nom que hi vulguem donar, en aquest cas *FRASE*, i a continuació la cadena. Per indicar la finalització d'introducció de text fareu servir el nom identificatiu que heu posat abans seguit de punt i coma (;). Heu de tenir en compte que el nom que doneu al bloc de text no es pot trobar dins el mateix text i sempre ha d'estar escrit al començament de la línia.

```
1 <?php
2 $nom='Abeeku';
3 $Familia='Informàtica';
4
5 echo <<<FRASE
6 Hola!<br>
7 Em dic $nom.
8 Estudio $Familia.
9 FRASE;
10 ?>
```

4. Utilitzant sintaxi *nowdoc*. A partir de la versió 5.3.0 del PHP s'inclou el suport per a *nowdoc*, per tant, no el feu servir si teniu una versió anterior. Aquesta sintaxi resulta molt útil quan heu de plasmar un bloc llarg de text que no necessita ser processat. És un mètode de treball amb cadenes molt ràpid i àgil. Per fer servir sintaxi *nowdoc* heu d'encabir el text entre dues marques especials. La marca d'inici és <<< seguida immediatament del nom que voleu posar escrit entre cometes simples. Per tancar el bloc de codi heu de tornar a escriure el nom identificatiu, però ara sense les cometes simples seguit de punt i coma (;). El nom que utilitzeu per marcar el bloc de text no ha d'aparèixer en el mateix text. En l'exemple següent, el valor de les variables no apareixeria en el navegador.

```
1 <?php
2 $nom='Abeeku';
3 $Familia='Informàtica';
4
5 echo <<<'FRASE'
6 Hola!<br>
7 Em dic $nom.
8 Estudio $Familia.
9 FRASE;
10 ?>
```

Matrius

Les matrius resulten molt útils en l'àmbit de treball del llenguatge PHP. De fet, en tots els llenguatges de programació les matrius són una peça fonamental, sense la qual seria molt difícil resoldre determinades situacions.

Una **matriu** és una col·lecció de valors. Poden ser unidimensionals, bidimensionals i multidimensionals.

Vectors o matrius

Tot i que es parli de manera genèrica de matrius cal aclarir que les matrius són realment *arrays* bidimensionals. Els *arrays* unidimensionals són els vectors i els *arrays* multidimensionals són *arrays* de més de dues dimensions. Durant el desenvolupament de la matèria es treballarà generalment amb *arrays* d'una dimensió.

Per accedir als valors que els *arrays* contenen cal utilitzar el delimitador []. En l'exemple següent es mostra com es crea un vector:

```
1 $credit[4]="Fonaments de Programació";
2 $credit[5]="Implantació d'Aplicacions Web";
3 $credit[10]="Síntesi";
```

En aquest sentit PHP és un llenguatge més flexible i menys estricte i estructurat, que d'altres com ara C o Java. No cal que definiu l'*array* prèviament. Heu creat un vector, per tant, d'una dimensió que conté alguns dels noms de crèdits del cicle que esteu estudiant. Tot i que no s'ha utilitzat, el vector comença en la posició 0. Per poder-vos moure utilitzareu un subíndex indicat entre [].

L'*array* pot créixer dinàmicament, per exemple, si voleu adherir el crèdit 7 només cal que feu:

```
1 $credit[7]="Relacions en l'Àmbit de Treball";
```

Si no utilitzeu un subíndex per treballar amb el vector, aquest el generarà automàticament. En l'exemple següent es crea un vector que contindrà els dies de la setmana i els traurà per pantalla:

```
1 <?php
2 $dies[]="Dilluns";
3 $dies[]="Dimarts";
4 $dies[]="Dimecres";
5 $dies[]="Dijous";
6 $dies[]="Divendres";
7 $dies[]="Dissabte";
8 $dies[]="Diumenge";
9
10 for($i=0 ; $i<count($dies) ; $i++)
11 {
12     echo $dies[$i];
13     echo "<br>";
14 }
15
16 ?>
```

Count és una funció que retorna la longitud del vector.

Matrius associatives

Les matrius associatives permeten al programador una flexibilitat i velocitat en la generació de codi molt importants. En PHP les matrius associatives s'han convertit en una eina potent de la qual sempre es depèn.

Una **matriu associativa** és una matriu que permet accedir als valors que emmagatzema mitjançant un subíndex de tipus cadena. És a dir, és un *array* en el qual les posicions es poden especificar per *strings* (en comptes dels índexs).

En l'exemple següent es crea la fitxa d'un alumne sense utilitzar una matriu associativa:

```
1 $dades[0]="Omar";
2 $dades[1]="12345678A";
3 $dades[2]="1956";
4 $dades[3]="ASIX";
```

Segons l'exemple anterior, si voleu recuperar alguna dada haureu de recordar el subíndex que us retornarà el valor que necessiteu; per exemple, per mostrar el DNI haureu d'escriure:

```
1 echo $dades[1];
```

Les matrius associatives les utilitzareu moltíssim quan trebal·leu amb variables predefinides.

En les matrius associatives tot és més fàcil. L'exemple següent mostra com es pot crear una fitxa personal i treure el DNI per pantalla:

```
1 $dades['nom']="Omar";
2 $dades['dni']="12345678A";
3 $dades['naixement']="1956";
4 $dades['cicle']="ASIX";
5
6 echo $dades['dni'];
```

Tal com es pot veure és molt més fàcil recordar l'índex d'una posició per un nom que no per un número.

Hi ha una altra manera de generar matrius associatives. L'exemple següent té un resultat idèntic a l'anterior:

```
1 $dades=array('nom'=>"Omar",
2             'dni'=>"12345678A",
3             'naixement'=>"1956",
4             'cicle'=>"ASIX");
```

2.1.8 Variables

El PHP és un llenguatge que garanteix dinamisme a les vostres pàgines web. Poder fer que determinats valors controlats per vosaltres canviïn en funció del codi es tradueix en un abandonament de la rigidesa del codi.

Una **variable** és un identificador al qual podreu assignar un valor, valor que podrà canviar al llarg del temps.

Per representar una variable caldrà escriure el símbol del dòlar seguit pel nom que vulgueu posar. En el moment de donar nom a una variable heu de tenir en compte que es diferencien majúscules i minúscules i que haurà de començar amb una lletra o caràcter de subratllat.

En PHP, a diferència de molts altres llenguatges, no cal declarar una variable abans d'utilitzar-la. El símbol del dòlar és el que marca que es tracta d'una variable.

```
1 <?php
2 $Nom='Essien';
3 $Familia='Informàtica';
4 $Cicle="CFGS Administració de Sistemes Informàtics en la Xarxa";
5 $Modul = "Implantació d'Aplicacions Web";
6 $Durada= 180;
7 echo "Hola $nom!<br>
8 Benvingut als estudis d'$Familia!!!<br>
9 Cicle: $Cicle<br>
10 Mòdul: $Modul<br>
11 Durada: $Durada hores";
12 ?>
```

Com podeu veure, les variables poden adquirir qualsevol dels tipus suportats, i la variable `$Durada` emmagatzema l'enter 180, mentre que la resta de variables emmagatzemaran cadenes.

Les variables s'assignen per valor per defecte, però si voleu assignar un valor per referència ho haureu d'indicar fent servir el símbol `&` tot just davant de la variable.

2.1.9 Estructures de control i sentències

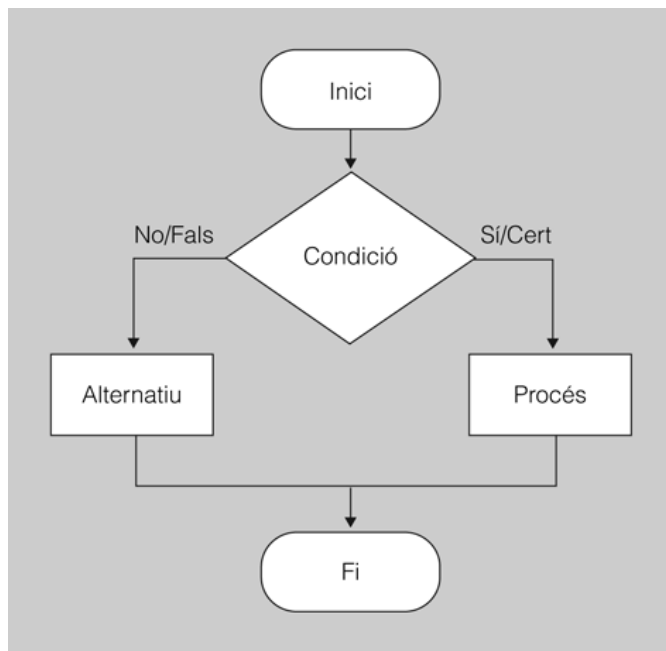
Quan comenceu a analitzar un problema i inicieu el procés de plasmar en un paper allò que us passa pel cap, de seguida us adoneu que necessiteu més eines, amb les sentències no en teniu prou per resoldre el problema.

Les **estructures de control** proporcionen dinamisme al vostre codi, i us brinden l'oportunitat d'indicar quines accions podeu prendre en determinades situacions.

El PHP us permet utilitzar un ampli ventall d'estructures de control, les quals són comunes en la gran majoria de llenguatges de programació. Si teniu més d'una sentència per executar dins una estructura, les haureu d'agrupar entre claus. Si només heu d'executar una sentència dins les estructures, l'ús de les claus és opcional.

Estructura de control: if, else, elseif

Mitjançant l'estructura *if* podreu condicionar l'execució d'un codi. En cas que la condició que s'analitza sigui certa, s'executarà un bloc de codi indicat. En la figura 3.2 podeu veure el diagrama d'aquesta estructura.

FIGURA 2.2. Estructura if else

En el cas següent es condiciona l'execució d'un bloc de codi en funció del valor de la variable *\$nom*. Si *\$nom* és igual a *Rashid* s'executarà el codi que hi ha entre les claus.

```
1 <?php
2 $nom='Rashid';
3 if($nom=='Rashid')
4 {
5     echo"Estàs dins el codi per que es complia la condició";
6 }
7 ?>
```

De vegades, resulta molt útil controlar els casos en què no es compleix la condició donada i provocar que s'executi un codi alternatiu.

```
1 <?php
2 $ampolla='buida';
3
4 if($ampolla=='plena')
5 {
6     echo "L'ampolla és plena";
7 }
8 else
9 {
10    echo "L'ampolla és buida";
11 }
12 ?>
```

L'estructura *if* us pot donar més alternatives a l'hora de crear condicionants en el vostre codi. Per exemple, davant l'execució d'un codi alternatiu podeu provocar el compliment d'una condició.

```
1 <?php
2 $nom='Nafisa';
3 if($nom=='Nafisa')
4 {
5     echo "Estàs dins el codi per què es compleix la condició";
6 }
```

```
7 elseif($nom=='Layla')
8 {
9     echo "Estàs dins el codi de la segona condició";
10 }
11 else
12 {
13     echo "No es compleix cap de les dues condicions";
14 }
15 ?>
```

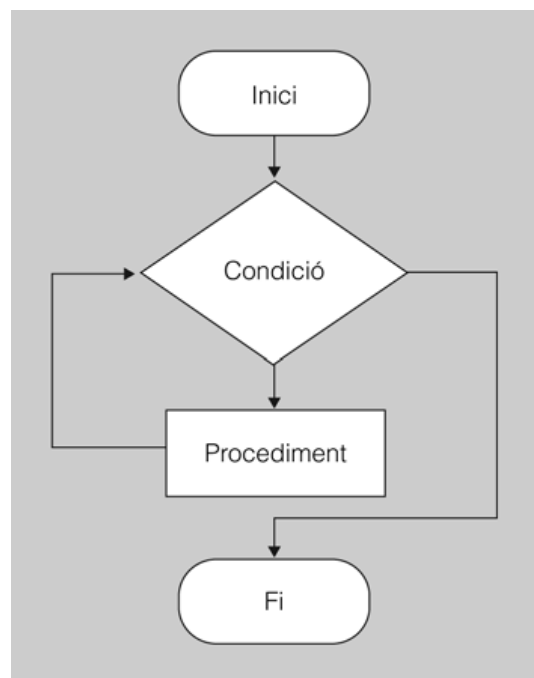
Igual que en la majoria de llenguatges de programació, podeu aniuar-hi codi.

```
1 <?php
2 $plou='si';
3 $paraigua='no';
4
5 if($plou=='si')
6 {
7     if($paraigua=='si')
8         echo "No em mullo!";
9     else
10         echo "M'estic mullant perquè no porto paraigua!";
11 }
12 else
13     echo "No cal que porti paraigua perquè no plou.";
14 ?>
```

Estructura de control: while

Fareu servir l'estructura *while* si us trobeu amb la necessitat de repetir una seqüència d'instruccions durant un període de temps controlat per una variable. En la figura 2.3 podreu veure el diagrama d'aquesta estructura.

FIGURA 2.3. L'estructura while



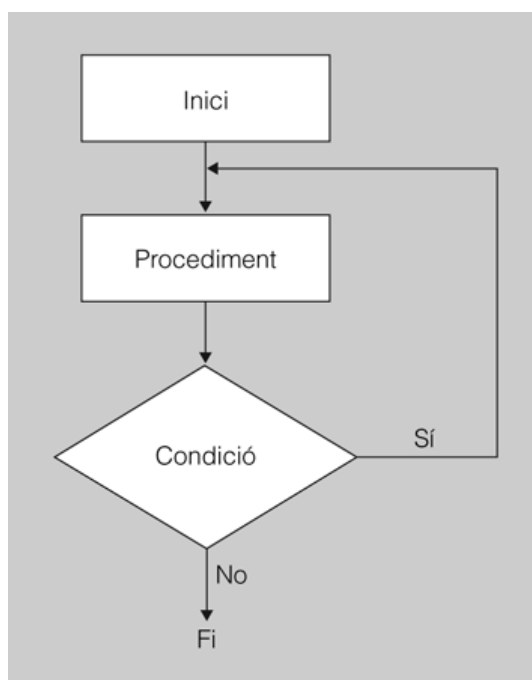
L'exemple següent mostra com el *while* us pot ser molt útil per crear un codi que generi la taula de multiplicar del número 7:

```
1 <?php
2 $multiplicand=7;
3 $multiplicador=0;
4 $resultat=0;
5 echo "Taula del $multiplicand:<br>";
6 while($multiplicador<=10)
7 {
8     $resultat=$multiplicand*$multiplicador;
9     echo"$multiplicand x $multiplicador = $resultat<br>";
10    $multiplicador=$multiplicador+1;
11 }
12 ?>
```

Estructura de control: do while

L'estructura *do* la utilitzareu en els casos en què sempre s'hagin de fer unes accions i, en finalitzar aquestes accions, es comprovi un estat que deixarà pas a una repetició del codi anterior o bé forçarà la sortida del bucle. En la figura 2.4 podeu veure el diagrama d'aquesta estructura.

FIGURA 2.4. L'estructura do



La sintaxi de l'estructura *do* inclou un *while*, que s'encarrega de comprovar una expressió.

En l'exemple següent es mostra en el navegador el número de línia corresponent mentre no arribem a la línia 10. Fixeu-vos que la comprovació del valor de la variable es fa una vegada s'ha produït una iteració.

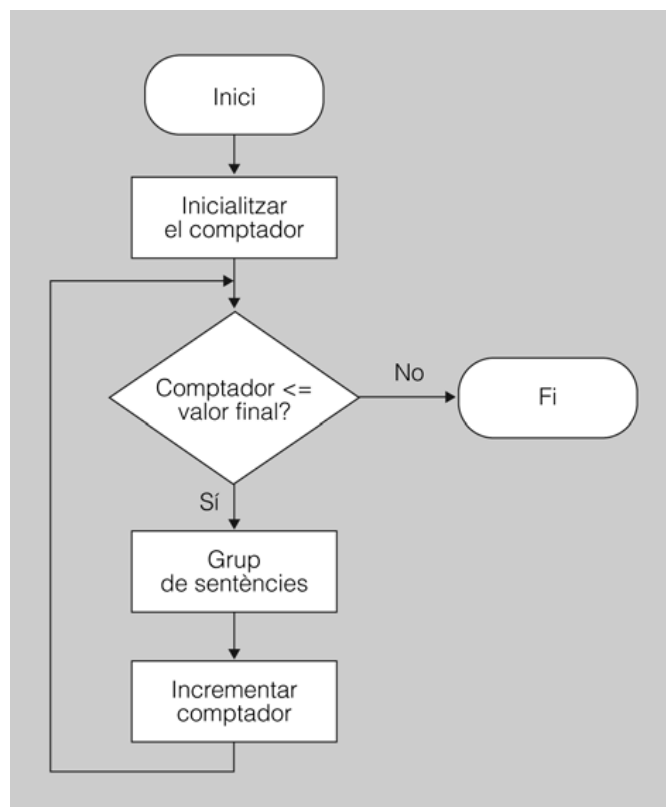
```
1 <?php
2 $linia=1;
3 do
4 {
5     echo "Línia $linia<br>";
6     $linia=$linia+1;
```

```
7 }  
8 while($linia<10);  
9 ?>
```

Estructura de control: for

L'estructura *for* utilitza tres expressions en la seva implementació. La primera expressió s'executa en iniciar-se el codi i dóna un valor inicial a una variable. Cal remarcar que aquesta variable tindrà un paper important en el desenvolupament del codi, serà la variable que permetrà el control del bucle. La segona expressió és la condició que ha de ser favorable per poder executar el bucle de codi. La tercera expressió indica una acció que cal dur a terme quan finalitzi el bloc de codi. En la figura 2.5 podeu veure el diagrama d'aquesta estructura.

FIGURA 2.5. L'estructura for



En l'exemple següent la variable *\$edat* s'inicialitza amb valor 0 i, mentre tingui un valor més petit que 18, es permetrà l'execució del bloc de codi que hi ha sota el control del *for*. Quan finalitzi el bucle, la variable *\$edat* s'incrementarà en una unitat.

```
1 <?php  
2 for($edat=0 ; $edat<18 ; $edat=$edat+1)  
3 {  
4     echo "Encara ets menor d'edat, ja que tens $edat anys!<br>";  
5 }  
6 echo "Ja no ets menor d'edat!";  
7 ?>
```

Qualsevol de les tres expressions que formen part de l'estructura *for* poden ser formades per més d'una expressió i, fins i tot, poden restar buides.

No cal que les tres expressions del *for* tinguin valor, les podeu deixar buides (però respectant els punt i coma). L'exemple següent mostrarà una sortida idèntica a l'exemple anterior, però ara dins el *for* només hi ha una expressió. La declaració i inicialització de la variable s'ha fet fora del bucle i, a més a més, augmentareu una unitat la variable *\$edat* dins el mateix codi del bucle.

```
1 <?php
2 $edat=0;
3 for( ; $edat<18 ; )
4 {
5     echo "Encara ets menor d'edat, ja que tens $edat anys!<br>";
6     $edat=$edat+1;
7 }
8 echo "Ja no ets menor d'edat!";
9 ?>
```

La sentència break

La sentència *break* provoca la sortida de qualsevol estructura de control, trencant així un bucle. Penseu per exemple un bucle que cerca un element d'un *array*. Una vegada trobat, és ineficient acabar de recórrer la resta de l'*array*.

Podeu indicar a aquesta sentència quantes estructures de control voleu saltar.

En l'exemple següent el *break* provoca la sortida de dos blocs de codi:

```
1 <?php
2 while($edat<20)
3 {
4     for( ; ; $edat=$edat+1)
5     {
6         if($edat==18)
7         {
8             echo "Ara ja tens $edat anys!<br>";
9             break 2;
10        }
11
12        echo "Encara ets menor d'edat, ja que tens $edat anys!<br>";
13    }
14
15    echo "Aquesta línia no sortirà per pantalla!";
16 }
17
18 echo "Ja no ets menor d'edat! Tens $edat anys!";
19 ?>
```

La sentència continue

La sentència *continue* provoca el salt de les línies de codi que restin per executar dins un bucle i continua executant la primera línia de codi que es trobi fora del bucle.

És possible indicar quants bucles se saltarà aquesta sentència.

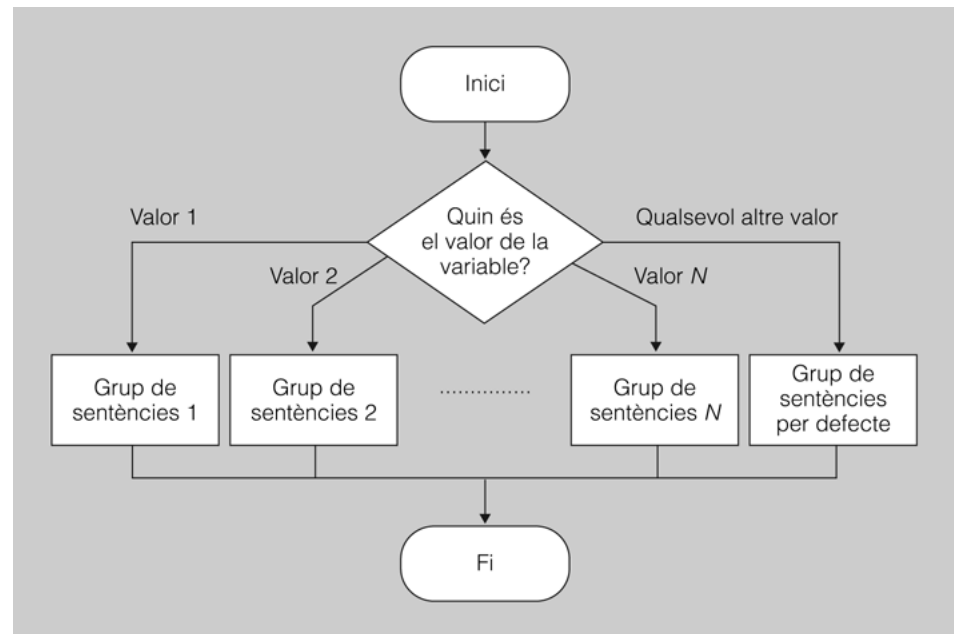
En l'exemple següent, la *continue* forçarà la sortida del bloc de codi encapçalat per *if*.

```
1 <?php
2 $edat=0;
3 $nom='Minna';
4 while($edat<=21)
5 {
6     if($edat==21)
7     {
8         echo "<br>Felicitats! Ja tens $edat anys, $nom! ";
9         echo "Tens el carnet adequat?";
10        $edat=$edat+1;
11        continue;
12        echo "Aquesta línia mai s'executarà";
13    }
14
15    echo "Ara tens $edat anys! Si no tens 21 anys no
16        podràs conduir una moto de gran cilindrada, $nom!<br>";
17    $edat=$edat+1;
18 }
19 ?>
```

L'estructura switch

L'estructura de control *switch* executa un codi determinat en funció del valor d'una variable. En la figura 2.6 podeu veure el diagrama d'aquesta estructura.

FIGURA 2.6. L'estructura switch



Per implementar l'estructura *switch* caldrà fer servir la sentència *case* seguida de dos punts, la qual s'encarregarà de comparar el valor de la variable que utilitza el *switch* amb el valor que identifica un codi determinat, i la sentència *break*, que forçarà la sortida del bloc de codi. Opcionalment es podrà utilitzar la sentència *default* seguida de dos punts, que ens permetrà executar un codi en cas que no es compleixi cap condició certa dins el *switch*.

En l'exemple següent l'estructura *switch* utilitza la variable *\$opcio*. En cas que la variable *\$opcio* valgui 0 sortirà un missatge per pantalla, si el valor d'opció fos 1 el missatge seria un altre, i si fos 2 el missatge també seria diferent; ara bé, en qualsevol altre cas s'executarà el codi que es trobi a continuació del *default*. En el cas concret de l'exemple, com que *\$opcio* té un valor de 3 s'executarà el codi del *default*.

```
1 <?php
2 $opcio=3;
3 switch($opcio)
4 {
5     case 0:
6         echo "Estàs aquí perquè el valor de la variable \$opcio és $opcio";
7         break;
8     case 1:
9         echo "Estàs aquí perquè el valor de la variable \$opcio és $opcio";
10        break;
11    case 2:
12        echo "Estàs aquí perquè el valor de la variable \$opcio és $opcio";
13        break;
14    default:
15        echo "Estàs aquí perquè no hi ha coincidència entre
16        el valor de la variable \$opcio ($opcio) i els valors dels case";
17 }
18 ?>
```

L'estructura foreach

L'estructura de control *foreach* permet recórrer una matriu. Aquesta estructura es pot fer servir a partir de la versió 4 del PHP.

En l'exemple següent es crea una matriu que emmagatzema quatre nombres. Amb l'estructura *foreach* es recorre la matriu posició a posició i es copien els nombres en la variable *\$valor*:

```
1 <?php
2
3 $matriu = array(1, 2, 3, 4);
4
5 foreach ($matriu as $valor)
6 {
7     echo "El valor de \$matriu és $valor.<br>";
8 }
9 ?>
```

També es pot assignar una clau a cada element. En l'exemple següent els índexs clau de la matriu són noms que emmagatzemen edats:

```
1 <?php
2 $matriu = array(
3     "Andreu" => 31,
4     "Marina" => 28,
5     "Antoni" => 19,
6     "Carla"  => 25
7 );
8
9 foreach($matriu as $posicio => $valor)
10 {
11     echo "L'alumne $posicio té $valor anys.<br>";
12 }
13 ?>
```

2.1.10 Funcions

En el desenvolupament normal d'un programa sempre arriba un punt en què es veu la necessitat d'agrupar codi i donar-hi un nom, per controlar-lo o per evitar repetir-ne contínuament les mateixes línies.

Una **funció** és un grup de codi independent que fa una tasca per al programa.

La funció executa l'operació i retorna el control del programa a la instrucció següent a la que la va cridar.

Funcions definides per l'usuari

L'usuari pot definir totes les funcions que necessiti. És imprescindible que el nom donat a la funció no estigui prohibit, és a dir, no es poden fer servir paraules reservades (per exemple, *echo*), començar amb un nombre o un caràcter especial ni repetir el nom d'una funció dins un mateix àmbit. Tot i que el PHP no farà cap distinció entre majúscules i minúscules, es recomana utilitzar el nom de la funció idèntic a l'utilitzat en la declaració.

Les funcions poden rebre paràmetres. El pas de paràmetres d'entrada es fa escrivint entre parèntesis i a continuació del nom de la funció.

En l'exemple següent podeu veure com es declara la funció *missatge* i com, posteriorment, es crida passant-hi un paràmetre.

```
1 <?php
2 function missatge($argument)
3 {
4     echo "$argument<br>";
5 }
6
7 missatge("Aquest és l'argument que es passa en aquest exemple.");
8 ?>
```

Les funcions també podran retornar valors. Per fer-ho caldrà utilitzar la sentència *return*.

En l'exemple següent comprovareu com la funció *suma*, la qual rep dos paràmetres, retorna el resultat de sumar dos nombres:

```
1 <?php
2 function suma ($valor1,$valor2)
3 {
4     return($valor1+$valor2);
5 }
6 echo suma(4,6);
7 ?>
```


Funcions internes

El llenguatge PHP té funcions definides internament. Aquestes funcions són de gran utilitat per al programador i resulta imprescindible estudiarles i conèixer-ne les més importants.

És possible que diverses funcions no es reconeguin en el vostre entorn de treball del PHP; el motiu d'això és que algunes funcions internes no s'han inclòs en la compilació, la qual cosa ha fet necessària una compilació dels mòduls que les contenen. Amb *phpinfo()* es poden veure les extensions carregades en la vostra configuració del PHP.

Són funcions internes molt usuals *echo*, *fprintf*, *print*, *printf* i *sprintf*, en què queda demostrat que el llenguatge PHP ja té definides funcions específiques que poden ser de gran utilitat per a l'usuari.

En l'exemple següent es mostra com es pot codificar amb codificació md5 el contingut d'una variable:

```
1 <?php
2     $nom = 'Blanca';
3
4     echo("La codificació md5 de $nom és:<br>");
5     echo md5($nom);
6 ?>
```

Podeu mirar l'ajuda de PHP a www.php.net/manual/es/funcref.php. Aquí podeu trobar una caixa per cercar la referència de les diferents funcions del llenguatge.

2.1.11 Classes i objectes

El llenguatge de programació PHP està preparat per fer ús d'aspectes propis de la programació orientada a objectes, tot i que el PHP no és un llenguatge de programació orientada a objectes.

Per definir una classe en llenguatge PHP s'ha d'utilitzar la paraula reservada *class* seguida del nom que hi voleu donar. Entre claus s'han d'ubicar els atributs i els mètodes propis de la classe.

Per instanciar objectes en PHP es farà servir l'operador *new* seguit del mètode constructor de la classe.

L'exemple següent mostra el funcionament de la classe Alumne. Aquesta classe té dos atributs que resumeixen les característiques de la classe. Podreu observar que els atributs *\$nom* i *\$curs* són precedits per l'operador *var*, el qual identifica com a variable el text que ve a continuació.

La classe Alumne conté un mètode especial anomenat *constructor*, el qual té la missió d'inicialitzar objectes. És molt fàcil identificar els mètodes constructors, ja que reben el mateix nom que la classe. Dins el mètode constructor s'assignen valors als atributs *\$nom* i *\$curs*. En aquest exemple s'utilitza *\$this*, que és una variable predefinida que fa referència a l'objecte que en aquest moment s'està utilitzant.

Simula 67

Es tracta d'un llenguatge creat per fer simulacions. Se'l considera l'origen de la programació orientada a objectes. Els seus autors són els noruecs Ole-Johan Dahl i Kristen Nygaard.

Una altra manera de generar mètodes constructors és utilitzar *function __construct*. Últimament aquesta variant és la que més s'utilitza, tot i que les dues fórmules són perfectament vàlides.

Podeu veure que dins la classe hi ha un mètode anomenat *avaluar* que s'encarrega d'assignar un valor a l'atribut *\$nota*.

Fora de l'àmbit de la funció es crea un objecte de la classe Alumne amb la partícula *new* seguida del mètode constructor.

```
1 <?php
2 class Alumne
3 {
4     var $nom;
5     var $curs;
6     var $nota;
7
8     function Alumne($nom,$curs)
9     {
10         $this->nom=$nom;
11         $this->curs=$curs;
12     }
13
14     function avaluar($nota)
15     {
16         $this->nota=$nota;
17     }
18 }
19
20 $nouAlumne=new Alumne('Laura',2);
21 echo "Nom de l'alumne: $nouAlumne->nom<br>";
22 echo "Curs: $nouAlumne->curs<br>";
23 $nouAlumne->avaluar(7);
24 echo "Nota: $nouAlumne->nota";
25 ?>
```

2.2 Integració PHP amb HTML

Normalment trobareu el llenguatge de programació PHP dins d'altres llenguatges. El més usual és escriure codi en PHP dins de codi HTML.

Heu de recordar que el codi PHP és interpretat pel servidor, i els resultats es mostren en el navegador que utilitzeu. La simbiosi que es crea entre el PHP i l'HTML permet al PHP generar HTML i a l'HTML passar dades al PHP.

Per començar a dominar el llenguatge PHP és recomanable que comenceu creant una pàgina bàsica. A partir d'aquesta pàgina podeu provar coses i construir les bases que necessitareu més endavant. Això és important, ja que l'aprenentatge de qualsevol llenguatge de programació comença en establir unes bones bases.

Heu de saber que no és necessari treballar amb una pàgina HTML per utilitzar el PHP; ara bé, per veure resultats en un navegador haureu de filtrar el PHP per un llenguatge que el navegador pugui interpretar, com l'HTML.

2.2.1 L'ús del PHP en formularis

El formulari electrònic és una eina molt potent fortament establida a Internet. Permet l'intercanvi o la introducció de dades entre usuaris d'una manera ràpida i senzilla. Un formulari permet la interactivitat i és el detonant que converteix Internet en l'eina més poderosa per enviar informació a un destinatari.

El llenguatge PHP representa un salt qualitatiu enorme en l'ús de formularis. L'HTML no és un llenguatge creat pensant a utilitzar formularis, sinó que es va crear amb altres objectius. El llenguatge PHP té un gran potencial en la gestió de dades; per tant, quan HTML i PHP treballen juntament amb formularis ens trobem amb un binomi ideal.

Preparació de l'HTML

El llenguatge HTML mostrarà la façana que l'usuari veurà a la pantalla del navegador. El PHP serà darrere d'aquesta primera pantalla i treballarà amb les dades.

En la figura 2.7 podeu veure l'aspecte típic d'un formulari. El codi necessari per dissenyar aquest formulari és elaborat en HTML; en canvi, el motor que espera les dades és constituït amb PHP.

FIGURA 2.7. Aspecte d'un formulari

Formulari per registrar un nou usuari en el sistema

* Camps obligatoris

Identificador d'usuari: *

Nom complet: *

Direcció de correu electrònica: *

Comentaris:

El primer que cal programar per fer servir un formulari dissenyat en PHP és crear el codi HTML. Un formulari en HTML s'emmarca entre les etiquetes `<FORM>` i `</FORM>`. Per manipular aquest formulari s'utilitza l'atribut ACTION.

PhpMyAdmin

L'eina de programari lliure PhpMyAdmin està creada íntegrament en llenguatge de programació PHP. És un exemple del que podeu arribar a programar amb aquest llenguatge de programació.

L'atribut `ACTION` indica el programa que s'encarrega de manipular les dades, tasca que encomanareu a un codi dissenyat en llenguatge PHP.

L'etiqueta `<form>` us permet fer servir dos mètodes per enviar informació: GET i POST. El mètode GET és molt útil en motors de cerca, però no ofereix gaire seguretat, ja que les dades que envia es poden veure en l'URL del navegador. El mètode POST és molt més segur, ja que quan el servidor detecta que el mètode de transacció d'informació és POST espera rebre dades immediatament sense fer servir l'URL, la qual cosa implica que aquestes dades no seran visibles en l'URL.

Un petit exemple de crida a un formulari seria:

```

1 <HTML>
2   <head>
3     <title>El nostre primer exemple amb formularis</title>
4   </head>
5   <body>
6     <H1>Introdueix el teu nom i prem el botó!</H1>
7
8     <FORM ACTION="contesta.php" METHOD="POST">
9       Si us plau, escriu el teu nom:<INPUT TYPE="text" NAME="nom"><BR>
10      <INPUT TYPE="submit" VALUE="Enviar">
11    </FORM>
12  </body>
13 </HTML>

```

Podeu veure com, dins l'etiqueta `<BODY>`, apareix l'etiqueta `<FORM>`. L'atribut `ACTION` crida un fitxer anomenat `contesta.php`, el qual contindrà el codi PHP encarregat de valorar el formulari. El pas de les dades es farà mitjançant el mètode POST. El servidor detecta que s'està executant un codi que conté transacció d'informació mitjançant el mètode POST; per tant, el servidor es posa a escoltar a l'espera que hi arribin dades. Les dades les rebirà directament, sense fer servir l'URL. En el moment en què es premi el botó etiquetat com a *Enviar* totes les dades introduïdes en el formulari s'envien al fitxer `contesta.php`.

Com valora el PHP un formulari

En el moment d'enviar les dades des del formulari d'una pàgina HTML a un fitxer escrit en PHP s'ha de tenir en compte que l'atribut `NAME` emmagatzema el valor.

HTML i PHP

En el treball de formularis i PHP és interessant que exploreu el món de les etiquetes de l'HTML. Si no sabeu com l'HTML treballa amb els formularis, limitareu les possibilitats del vostre codi.

```

1 <HTML>
2   <head>
3     <title>Resposta a la petició de dades</title>
4   </head>
5   <body>
6     <H1>...Continuem amb l'exemple!</H1>
7     El teu nom és:
8     <?php
9       echo $_POST[nom];
10    ?>
11   </body>
12 </HTML>

```

En el moment en què s'envien dades del formulari HTML cap a un fitxer PHP, aquestes dades apareixeran dins d'un `array` superglobal amb tots els seus valors.

Les matrius associatives i els formularis

Resulta de gran utilitat fer servir matrius associatives per tractar la informació rebuda d'un formulari.

Podeu utilitzar les matrius associatives `$_POST` i `$_GET`; el seu ús, però, serà condicionat pel mètode utilitzat en el pas de dades en el codi HTML. Així, si en el codi HTML el mètode és POST haureu de fer servir la matriu associativa `$_POST`; en canvi, si heu fet servir el mètode GET en el codi HTML, haureu de fer servir la matriu associativa `$_GET`.

Per accedir als valors emmagatzemats en les matrius associatives fareu servir l'atribut corresponent NAME de l'HTML com a índex de la matriu associativa.

No heu d'oblidar que el PHP és un llenguatge de programació que, entre altres coses, us permet utilitzar bucles. En el formulari següent es defineix una selecció múltiple, en què podreu marcar els navegadors que coneixeu.

```

1 <HTML>
2   <head>
3     <title>Un exemple amb un formulari</title>
4   </head>
5   <body>
6     <form action="seleccioMultiple.php" method="POST">
7       Nom: <input type="text" name="nom"><br>
8       Correu electrònic: <input type="text" name="email"> <br>
9       Navegadors que coneixes: <br>
10      <select name="navegadors[]" multiple>
11        <option value="Chrome">Chrome</option>
12        <option value="MozillaFirefox">Mozilla Firefox</option>
13        <option value="Safari">Safari</option>
14        <option value="Opera">Opera</option>
15        <option value="Lynx">Lynx</option>
16      </select><br>
17      <input type="submit" value="Envia" >
18    </form>
19  </body>
20 </HTML>

```

Aquí us trobareu amb un petit problema. Per recuperar els valors introduïts com a nom i correu electrònic no hi ha cap problema, la matriu `$_POST` us permetrà un recorregut buscant els valors emmagatzemats a *nom* i *email*. Amb la selecció múltiple, però, sorgeix un petit problema. A diferència dels casos anteriors, ara *navegadors* pot contenir més d'un valor, com ho fareu per mostrar tots els valors per pantalla? La solució és utilitzar un bucle:

```

1 <?php
2 foreach ($_POST['navegadors'] as $opcio)
3   echo $opcio."<br>";
4
5 ?>

```

El bucle *foreach* anterior permet recórrer la matriu *navegadors*, la qual contindrà els navegadors que heu marcat en el formulari inicial.

Si no recorreu la matriu i la mostreu directament amb:

```

1 echo $_POST['navegadors'];

```

només es mostrarà la cadena *Array*.

2.2.2 Tipus de dades en PHP

En PHP no és necessari declarar una variable abans d'utilitzar-la; de fet, no cal ni establir el tipus al qual pertany, el mateix llenguatge s'encarregarà d'assignar el tipus corresponent segons el contingut i el context. Fixeu-vos en l'exemple següent:

```
1 <?php
2 $valor=123;
3 echo "$valor<br>";
4 $valor='Hola';
5 echo "$valor<br>";
6 $valor=123+'Hola';
7 echo "$valor<br>";
8 ?>
```

Podeu veure que en l'exemple anterior en cap moment no s'ha definit el tipus de la variable.

Si voleu, podeu indicar el tipus de la variable o forçar una conversió de tipus. Una utilitat que fareu servir sovint és forçar el tipus d'una variable per validar l'entrada de dades.

El llenguatge del PHP suporta vuit tipus primitius, dos d'especials i alguns pseudotipus.

Tipus de dades primitius

Els tipus de dades primitius suportats pel PHP són vuit, agrupats en escalars, compostos i especials. Per dinamitzar el tractament de dades i ajudar a comprendre alguns codis es permet utilitzar pseudotipus, com ara *mixed*, *number* o *callback*.

1. Els tipus de **dades escalars** són:

- *boolean*: és un tipus de dades molt simple que pot representar un valor cert (*TRUE*) o fals (*FALSE*). Aquest tipus de dades s'utilitza molt en estructures de control. Si feu conversió de dades és molt important que tingueu en compte que es considerarà fals el nombre enter i amb coma 0, la cadena 0 o una cadena buida, una matriu buida, un objecte sense variables membre, el tipus

NULL i els objectes *SimpleXML* que s'hagin creat d'etiquetes buides. A continuació, es mostren dues variables booleans:

```
1 $sortir=FALSE;
2 $entrar=TRUE;
```

- *integer*: aquest tipus representa nombres no fraccionables per sota de la unitat. Poden ser positius i negatius. Aquests nombres enters els podeu representar en notació decimal, hexadecimal o octal. La plataforma sobre la qual treballem determinarà la mida de les variables enters. El PHP forçarà un canvi de tipus a flotant si es produeix un desbordament, tant si és positiu com negatiu. A continuació, es mostren dos exemples de variables enters:

```
1 $positiu=3451;  
2 $negatiu=-4345;
```

- *float*: aquest tipus fa referència als nombres de coma flotant, i normalment treballa amb una precisió aproximada de catorze dígitos decimals. A continuació, es mostra un exemple d'una variable amb coma flotant:

```
1 $amb_coma=34.5;
```

- *string*: es tracta d'una cadena de caràcters. El llenguatge del PHP no imposa un límit a la mida d'aquestes cadenes; ara bé, no oblideu que el límit el marcarà la memòria del maquinari en què estigui funcionant el PHP. Aquí teniu diversos exemples de cadenes:

```
1 <?php  
2 $nom='Wayan';  
3 $salutacio="Hola $nom!";  
4 echo $salutacio." Benvingut";  
5 ?>
```

En l'exemple anterior es fan servir cometes dobles i simples per marcar les cadenes. La diferència entre fer-ho d'una manera o d'una altra és que amb cometes dobles s'interpreten les variables com a tals, mentre que amb cometes simples no es mostrarà el contingut de les variables. Una manera de concatenar cadenes és utilitzar l'operador punt.

2. Els tipus de **dades compostos** són:

- *array*: un *array* és una matriu, d'una dimensió o més, que conté dades que s'associen segons unes claus. És un tipus molt útil, i comprovareu que el seu ús té una gran potència. Per crear una matriu podeu utilitzar la paraula reservada *array()* o bé acompanyar el nom de la variable amb *[]*.

En l'exemple següent es creen dos *arrays*, un que conté els dos graus de cicles formatius que hi ha, i un altre que emmagatzemarà dues famílies professionals. Podeu veure que en el segon cas la paraula *array* no apareix per enlloc i es crea la matriu igualment:

```
1 $grau_cicles=array("superior","mitja");  
2 $famílies[]="Informàtica";  
3 $famílies[]="Electrònica";
```

- *object*: aquest tipus d'objecte està previst en el llenguatge PHP perquè suporta el llenguatge orientat a objectes. Un objecte és un ítem amb unes característiques i funcionalitats marcades per la classe a la qual pertany. Per crear un objecte cal utilitzar la partícula *new*.

```
1 <?php
2 class Alumne {
3
4     public function matricular($nom)
5     {
6         echo "Ja estàs matriculat $nom";
7     }
8 }
9
10 $alumne1=new Alumne();
11 $alumne1->matricular('Mada');
12 ?>
```

3. Els tipus de **dades especials** són:

- *resource*: la variable que pertany a aquest tipus farà referència a un recurs extern. Determinades funcions faran ús d'aquest recurs. Per exemple, la funció *mysql_db_query*, que té la funció d'enviar una petició *MySQL*, retorna un tipus *resource*. En l'exemple següent la variable *\$resultat* és de tipus *resource*:

```
1 $query='SELECT * FROM alumnes';
2 $resultat=mysql_query($query);
```

- *NULL*: quan necessiteu una variable que no tingui cap valor haureu d'utilitzar aquest tipus.

4. Els **pseudotipus** són:

- *mixed*: aquest pseudotipus indica que la variable en qüestió pot emmagatzemar múltiples tipus.
- *number*: indica que la variable pot contenir un valor enter o de coma flotant.
- *callback*: utilitzareu aquest pseudotipus per emmagatzemar crides de retorn generades per funcions de l'usuari.

Manipulació de tipus

El llenguatge PHP fa la majoria de conversions de tipus d'una manera automàtica. Una conversió molt normal és la que es fa de cadenes a valors numèrics. Si, per exemple, la cadena conté un punt o la lletra *e* (majúscula o minúscula) i la sumeu a un enter, la cadena es considerarà un nombre de coma flotant, i el resultat serà de tipus de coma flotant.


```
1 <?php
2 $cadena="Aquí hi ha text";
3 echo 'Contingut de $cadena: '.$cadena."<br>";
4 $numero=12;
5 echo 'El valor de $numero: '.$numero."<br>";
6 echo 'Ara $cadena conté: '.$cadena=$cadena+$numero;
7 ?>
```

En l'exemple anterior la variable *\$cadena* passa de ser un *string* a un *integer*.

Conversió de tipus (casting)

Per forçar un canvi de tipus haureu d'especificar entre parèntesis el tipus al qual voleu fer la conversió i col·locar-lo davant la variable.

En la taula 2.8 podeu trobar les conversions permeses.

TAULA 2.8. Modelitzacions de tipus

Sentència	Modelat a...
(integer), (int)	integer
(boolean)	boolean
(float), (double), (real)	float
(string)	string
(binary)	string
(array)	array
(object)	object

En l'exemple següent es mostra com es pot utilitzar la conversió (*casting*). En aquest cas, es modela el que hauria de ser un nombre amb decimals a un nombre enter:

```
1 <?php
2 echo "Sense modelar 27/5: " . (27/5) . "<br>";
3 echo "I ara ja modelat: " . (integer)(27/5);
4 ?>
```

2.2.3 Variables

El llenguatge de programació PHP té una característica que destaca. El món de les variables en PHP és una gran ajuda per al programador, i això no ve al cas pel fet que no calgui declarar els tipus de les variables, sinó perquè té un gran nombre de variables predefinides que simplifiquen molt les tasques del tècnic.

Àmbit de les variables

Les variables tenen una vida determinada depenent d'on les declareu i com ho feu. És molt important conèixer l'àmbit de les variables per controlar els vostres codis. No comprendre aquesta part pot provocar fallades inesperades en les vostres creacions.

L'àmbit d'una variable és el context en què és definida.

En l'exemple següent es mostra com es pot declarar una variable nombroses vegades en àmbits diferents:

```
1 <?php
2 $nom='Nyoman';
3 echo 'De moment $nom val ' . $nom . "<br>";
4
5 function canvi()
6 {
7     $nom='Ketut';
8     echo 'Però ara $nom val ' . $nom . "<br>";
9 }
10
11 canvi();
12 echo 'I de nou $nom torna a tenir el valor de ' . $nom . "<br>";
13 ?>
```

Variables predefinides

En PHP hi ha una sèrie de **variables predefinides** que resulten de gran ajuda per al programador. Dins d'aquestes variables predefinides cal destacar un gran grup, el grup de les anomenades *superglobals*.

Les **variables superglobals** són variables que sempre estan disponibles en tots els àmbits.

Les variables superglobals són nou:

1. \$GLOBALS: fa referència a totes les variables disponibles en l'àmbit global. Es tracta d'una variable de tipus matriu associativa.

En l'exemple següent es recorre la variable superglobal **\$GLOBALS**:

```
1 <?php
2 $una_variable="Hola";
3 $una_altra_variable="Com va?";
4
5 foreach ($GLOBALS as $key => $valor)
6     echo "GLOBALS[$key] = $valor<br>";
7 ?>
```

2. \$_SERVER: ofereix informació molt útil sobre el servidor i l'entorn d'execució. Els valors que emmagatzema són generats pel navegador web.

```
1 <?php
2 foreach ($_SERVER as $key => $valor)
```

Una manera de recórrer matrius és utilitzar la funció *foreach*. Introduïda en el PHP4 s'ha revisat en el PHP5 per iterar també objectes.

```
3 echo "SERVER[$key] = $valor<br>";  
4 ?>
```

L'exemple anterior és de gran utilitat. Executeu-lo i veureu un seguit de dades referents a la màquina que fa de servidor del PHP. Entre altres coses, identificareu el nom de la màquina, la versió del navegador, la versió del servidor web, la ubicació de fitxers importants i un llarg etcètera.

3. `$_GET`: correspon a la variable `GET` del protocol HTTP. Emmagatzema els valors de les variables que es trasmeten per mitjà del protocol `HTTP` mitjançant el mètode `GET`. El seu ús és molt normal en formularis.

4. `$_POST`: correspon a la variable `POST` del protocol HTTP. Emmagatzema els valors de les variables que es transmeten per mitjà del protocol `HTTP` mitjançant el mètode `POST`. El seu ús és, generalment, més adequat que el mètode `GET` per a temes de seguretat.

5. `$_FILES`: aquesta variable s'utilitza en la càrrega de fitxers per mitjà del protocol HTTP. El mètode utilitzat és el `POST`.

6. `$_COOKIE`: és una variable de tipus matriu associativa que emmagatzemarà les variables passades a l'*script* per mitjà de les galetes de l'HTML.

7. `$_SESSION`: conté les variables de sessió disponibles en l'*script* actual.

8. `$_REQUEST`: és una matriu associativa que conté les dades de `$_GET`, `$_POST` i `$_COOKIE`.

9. `$_ENV`: conté les variables d'entorn.

Altres variables predefinides són:

- `$php_errormsg`: aquesta variable emmagatzema el text de l'últim missatge d'error. Per poder-la utilitzar, l'opció de configuració `track_errors` ha d'estar activada.
- `$HTTP_RAW_POST_DATA`: en aquesta variable quedaran emmagatzemades les dades `POST` originals.
- `$http_response_header`: conté les capçaleres de resposta HTTP.
- `$argc`: indica el nombre de paràmetres que s'han passat a l'*script*.
- `$argv`: conté els arguments que s'han passat a l'*script*.

Variables globals

La paraula reservada global indica que la variable que l'acompanya s'ha d'entendre en un àmbit global.

També podeu fer servir l'*array* associatiu `$GLOBALS`, que conté referències a totes les variables globals definides en cada moment en l'àmbit del programa. Els índexs de l'*array* són els noms de les variables.

En l'exemple següent es mostren tots dos casos, i podeu veure com es traspassen els problemes derivats de l'àmbit local de les variables.

```
1 <?php
2 $nom='Elewa';
3 $benvinguda='benvingut al curs!';
4
5 function salutacio()
6 {
7     global $nom, $benvinguda;
8
9     $benvinguda= $nom." ".$benvinguda."<br>";
10 }
11
12 salutacio();
13 echo $benvinguda;
14
15 echo "Valor d'una variable global: ".$GLOBALS['nom']."<br>";
16 echo "i de l'altra: ".$GLOBALS['benvinguda']."<br>";
17 ?>
```

Quan dins la funció *salutacio()* es declaren globals les variables *\$nom* i *\$benvinguda*, es força que totes les referències a aquestes dues variables apuntin a les variables globals.

Variables estàtiques

Les **variables estàtiques** sempre mantenen el valor encara que el programa abandoni l'àmbit de la variable, i existeixen només en l'àmbit local de les funcions. No s'hi pot accedir fora d'aquest àmbit.

Fixeu-vos en l'exemple següent. Hi ha dues funcions, una de les quals conté una variable estàtica, mentre que l'altra no. L'acció de totes dues funcions no és cap altra que incrementar el valor d'una variable, però els resultats són molt diferents. En la funció anomenada *comptador()* la variable que s'incrementa no és estàtica; per tant, cada vegada que la seqüència del programa abandoni la funció, la variable perdrà el seu valor. En canvi, en la funció *comptadorEstatic()*, la variable que s'incrementa és estàtica; per tant, quan la seqüència del programa abandoni aquesta funció no es perdrà el valor acumulat, i aquest restarà en la memòria.

```
1 <?php
2 function comptador()
3 {
4     $a = 0;
5     echo "Al comptador: ".$a."<br>";
6     $a++;
7 }
8
9 function comptadorEstatic()
10 {
11     static $b = 0;
12     echo "Al comptador estàtic: ".$b."<br>";
13     $b++;
14 }
15
16 for($i=0;$i<3;$i++)
17 {
18     comptador();
19     comptadorEstatic();
20 }
21 ?>
```

Variables variables

De vegades, pot resultar de gran utilitat disposar de noms de variables que es puguin definir i utilitzar dinàmicament; d'aquestes, en diem **variables variables**. Mireu l'exemple següent, en què una variable variable pren el valor d'una variable i el tracta com si fos el nom d'una variable.

```
1 <?php
2 // no podeu deixar espais dins de la cadena Hola, doncs es farà servir per nom
   d'altra variable
3 $variable = "Hola";
4
5 $$variable = " mon";
6
7 //Mostra el mateix
8 print " $variable{$$variable}<br>";
9 print " $variable$Hola<br>";
10
11 ?>
```

No podreu utilitzar les variables superglobals com a variables variables dins de funcions o mètodes de classe.

Variables de sessió

Les necessitats actuals forcen a buscar solucions ràpides i segures per accelerar la presentació d'informació i la comunicació entre les diferents parts que formen un sistema informàtic comunicatiu. Les variables de sessió són una solució molt adequada per resoldre aquests problemes.

Les **variables de sessió** són variables que estan disponibles en múltiples pàgines sense haver de fer servir pas de paràmetres. Les variables de sessió tenen un temps de vida limitat i s'emmagatzemen en el servidor.

Podeu fer servir les variables de sessió per mantenir informació (l'anomenada *informació de sessió*) entre diferents pàgines. Per exemple, el nom d'usuari o una selecció de productes que ja s'hagi fet.

A continuació, estudiareu un exemple format per tres fitxers. El primer fitxer és un formulari senzill que demana a l'usuari un identificador i una contrasenya. Aquestes dades s'envien a un segon fitxer, anomenat `exempleAutenticar.php`, que crea les variables de sessió. Finalment, trobareu un tercer fitxer anomenat `continuem.php`, que recupera la informació emmagatzemada en les variables de sessió.

Just a sota d'aquest text hi ha el fitxer del qual partireu. L'hem anomenat `exempleSessions.php`, però el podeu anomenar d'una altra manera:

```
1 <HTML>
2   <head>
3     <title>Exemple amb variables de sessió</title>
4   </head>
5   <body>
```

`<input type="password">`

Si indiqueu a l'etiqueta HTML que sigui de tipus contrasenya, no s'imprimiran els caràcters conforme aneu escrivint. En comptes de text es mostraran asteriscs.

```
6      <form action="exempleAutenticar.php" method="post">
7          Escriu el teu nom:
8          <input type="text" name="usuari"><br>
9
10         Escriu la teva contrasenya:
11         <input type="password" name="contrasenya"><br><br>
12
13         <input type="submit" value="Envia!">
14     </form>
15 </body>
16 </HTML>
```

Tot seguit apareix el codi del fitxer `exempleAutenticar.php`. En aquest fitxer creareu i inicialitzareu les dues variables de sessió que utilitzareu.

```
1 <?php
2 session_start();
3 $_SESSION['id_usuari']=$_REQUEST['usuari'];
4 $_SESSION['clau_acces']=$_REQUEST['contrasenya'];
5 ?>
6
7 <HTML>
8     <head>
9         <title>...continueu amb l'exemple de variables de sessió...</title>
10    </head>
11    <body>
12        Ja s'han emmagatzemat les variables de sessió.<br><br>
13        <a href="continuem.php">Vinga! Cap al final de l'exemple!</a>
14    </body>
15 </HTML>
```

Fixeu-vos que el primer que es fa és cridar la funció `session_start()`. És obligatori que inicieu la sessió abans de cap etiqueta HTML perquè tot funcioni correctament.

Amb `$_REQUEST` es recuperen els dos valors que s'han passat des del fitxer inicial, és a dir, *usuari* i *contrasenya*, i els heu d'emmagatzemar en dues variables de sessió creades a `$_SESSION`, que rebran el nom d'*id_usuari* i *clau_acces*.

`$_REQUEST` és un *array* associatiu global que per defecte té els continguts de `$_GET`, `$_POST` i `$_COOKIE`.

A part de crear les variables de sessió i donar-los valors trobareu un enllaç cap al tercer fitxer.

El codi del tercer fitxer, `continuem.php`, el teniu a continuació:

```
1 <?php
2 session_start();
3 ?>
4
5 <HTML>
6
7 <head>
8     <title>...i acabem l'exemple de variables de sessió!</title>
9 </head>
10
11 <body>
12     <?php
13         echo "L'usuari que havíeu escrit era: ".$_SESSION['id_usuari'];echo "<br>";
14         echo "La contrasenya que havíeu escrit era: ".$_SESSION['clau_acces'];
15     ?>
16 </body>
17 </HTML>
```

Com que esteu treballant amb sessions, el primer que hauríeu de fer és llançar la sessió amb `session_start()`. A continuació, es recuperen els valors emmagatzemats en les dues variables de sessió.

2.2.4 Excepcions en PHP

No diu gaire el codi que, a causa d'un error produït en temps d'execució, fa caure tot el programa i en força la sortida.

Les **excepcions** tenen la missió de detectar i corregir errors.

En cas que es produís un error, la vostra aplicació no hauria de produir un terrabastall i forçar la sortida del programa, sinó que es pot llançar una excepció (*throw*) que haureu de capturar (*catch*) i resoldre la situació d'error. El codi susceptible de provocar l'error ha d'anar dins d'un bloc *try*, que obligatòriament ha de tenir un bloc *catch*.

Quan es produeix una excepció, el PHP va a buscar directament el bloc de codi del *catch*. Si no es captura una excepció, l'error resulta fatal per al programa.

En l'exemple següent es fa un filtratge perquè no hi hagi cap divisió en què el divisor tingui per valor 0. Si això passa, en comptes de caure el programa, es tracta el problema. En aquest cas, quan es dona la situació simplement es mostra un missatge per pantalla que n'avisa:

```
1 <?php
2 function filtreDividir($dividend,$divisor)
3 {
4     if ($divisor==0)
5         throw new Exception('Divisió per zero.');
```

```
6
7     else
8         return ($dividend/$divisor);
9 }
10
11 try
12 {
13     $dividend=$_POST['dividend'];
14     $divisor=$_POST['divisor'];
15     filtreDividir($dividend,$divisor);
16 }
17 catch (Exception $e)
18 {
19     echo "S'ha capturat l'excepció: ".$e->getMessage()."<br>";
20 }
21
22 echo "I no s'avorta l'execució del programa!!!";
23 ?>
```

Hi ha dues excepcions predefinides: *Exceptions* i *ErrorException*. La primera és la classe base de totes les excepcions, conté el missatge de l'excepció, el nom de l'excepció, el codi de l'excepció, el nom del fitxer que ha llançat l'excepció, el número de línia en què s'ha produït l'excepció i la traça associada a l'excepció. L'excepció *ErrorException* hereta de l'excepció *Exception* i indica numèricament la severitat de l'excepció produïda.

2.2.5 Galetes (cookies)

Quan navegueu per Internet visiteu pàgines que poden amagar una certa complexitat per donar un servei millor a l'usuari. Podeu comprovar com diverses pàgines web, fins i tot, us reconeixen en el moment en què hi accediu.

Lou Montulli

Aquest programador va crear les galetes mentre treballava a Netscape durant la dècada de 1990. També va crear, juntament amb altres dos programadors, el popular navegador Lynx.

Les **galetes** són uns fitxers de text que emmagatzemen informació a l'ordinador client referent a la visita d'un usuari a una pàgina web.

Les galetes no són un acte d'intromissió per part de la pàgina web visitada; simplement, són una font d'informació molt útil per augmentar l'eficiència en l'accés.

Les galetes tenen una mida màxima d'1K i caduquen, la qual cosa fa molt difícil que s'utilitzin amb fins malintencionats. A més, únicament pot consultar la galeta el web que l'ha dipositada, la informació que conté no és visible per a la resta de pàgines web.

Utilitat de les galetes

Un ús molt freqüent de les galetes és emmagatzemar un perfil d'usuari. Així s'aconsegueix que cada vegada que el mateix usuari accedeix a un web, aquest en conegui les preferències i li mostri directament els continguts més adequats.

Les galetes ajuden els programadors a crear un entorn més adequat per a l'usuari visitant. Moltes pàgines web permeten canviar la mida del text, els colors de la pàgina, la disposició d'alguns continguts, etc. Una vegada l'usuari ha modificat al seu gust l'aspecte de la pàgina web, aquestes dades s'emmagatzemen per a la propera visita.

Funcions del PHP per treballar amb galetes

La funció que s'utilitza en PHP per crear galetes és *setcookie*.

La definició de la funció *Setcookie* és:

```
bool setcookie ( string $nom [, string $valor [, int
$caduca= 0 [, string $ruta [, string $domini [, bool
$seguretat= false [, bool $nomeshttp= false ]]]]] )
```

La funció *setcookie* retornarà *cert* si la galeta s'ha creat sense cap dificultat i *fals* si hi ha hagut algun problema.

A continuació anem a descriure els seus paràmetres:

- *\$nom*: és obligatori i indica el nom que donareu a la galeta, la resta de paràmetres són opcionals.
- *\$valor*: és el valor que tindrà la galeta.
- *\$caduca*: indica els segons que té de vida la galeta; si no s'indica res la galeta deixarà d'existir quan finalitzi la sessió de l'usuari.
- *\$ruta*: indica on podeu trobar la galeta dins el servidor.
- *\$domini*: marca el domini en què la galeta està disponible.
- *\$seguretat*: tindrà el valor *TRUE* si voleu que la galeta només s'utilitzi en una connexió segura, i *FALSE* (que és el valor per defecte), si no és necessari un cert grau de seguretat.
- *\$nomeshttp*: si té per valor *TRUE* marcarà que la galeta solament és accessible si s'utilitza el protocol HTTP.

L'exemple següent mostra l'ús d'una galeta. L'exemple consta de dues parts, un primer exemple en què es crea la galeta i s'hi assigna un valor, i una segona part en què es recupera el valor emmagatzemat en la galeta.

Galeta és la traducció de l'anglès *cookie*. El nom originari és *magic cookie*, que són les galetes de la sort, les quals amaguen informació a l'interior.

El primer fitxer s'encarrega de declarar la galeta amb un identificatiu, en aquest cas *nom*, i s'hi assignarà el valor *Maadiva*:

```
1 <?php
2     setcookie('nom', 'Maadiva');
3 ?>
```

El segon fitxer s'encarregarà de consultar a la matriu associativa *\$_COOKIE* el valor de la galeta indicada, que en aquest cas seria *nom*:

```
1 <?php
2     echo $_COOKIE['nom'];
3 ?>
```

2.2.6 Encriptació de dades amb el PHP

La seguretat és un aspecte molt important que cal tenir en compte en l'administració d'un sistema. El llenguatge PHP us ofereix la possibilitat d'aplicar tècniques d'encriptació mitjançant funcions com *crypt()*, *md5()* i les funcions *mcrypt*.

A més d'encriptar dades podreu verificar les vostres contrasenyes amb funcions *crack*, utilitzar funcions *hash* per identificar fitxers o comprovar signatures amb OpenSSL.

SHA1

SHA1 és una funció de *hash* criptogràfica dissenyada per l'agència de seguretat nacional nordamericana. La informació està codificada amb una clau de 160 bits i es considera una funció criptogràfica més segura que MD5 (que va ser durant molt de temps la funció criptogràfica més popular a PHP).

La funció per calcular el *hash* SHA1 té la següent forma a PHP:

```
1 string sha1 ( string $str [, bool $sortida_raw = false ] )
```

Aquesta funció, calcula el *hash* de la variable *\$str* i el retorna en forma de números hexadecimals de 40 dígitos en format *string*. Si l'argument opcional *\$sortida_raw* està ficat a *TRUE* la sortida tindrà el format binari amb una mida de 20 bytes.

En el següent exemple podeu veure el resultat d'encriptar una paraula i com es comprova amb el seu valor encriptat.

```
1 <?php
2 $str = 'apple';
3
4 echo sha1($str);
5
6 if (sha1($str) === 'd0be2dc421be4fcd0172e5afceea3970e2f3d940')
7 {
8     echo "Has seleccionat una poma";
9 }
10 ?>
```

Tot i poder encriptar les contrasenyes amb SHA1 o MD5, aquests algorismes han estat dissenyats per ser ràpids i eficients. Per aquests motius, amb l'ús de força bruta aplicada mitjançant els ordinadors actuals, es poden trencar sense moltes complicacions aquestes encryptacions. Alguns experts en criptografia no les recomanen com a sistema d'encryptació segur per les contrasenyes.

Algorismes de hash

Si s'aplica un algorisme determinat sobre un fitxer es pot arribar a obtenir un segon fitxer. Aquest segon fitxer conté un *número resum* del fitxer original.

Si els continguts del fitxer canvien, també ho farà el número resultant de la funció de *hash*. D'aquesta manera es pot comprovar si el contingut d'un fitxer és el mateix o no després d'alguna transacció en concret (per exemple, després de descarregar-lo d'Internet o de fer-ne una còpia de seguretat).

L'algorisme de *hash* permet obtenir una sèrie de bits de longitud fixa a partir d'un fitxer qualsevol. És impossible la reconstrucció del fitxer original utilitzant el codi *hash* obtingut.

Les aplicacions més esteses del *hash* són:

- Comprovar la integritat dels fitxers.
- Reforçar la seguretat en processos d'autenticació.

- Signatura digital.

No confongueu l'algorisme *hash* amb les taules *hash*. Les taules *hash* són una estructura de dades.

Funcions hash

El llenguatge PHP suporta diferents funcions que utilitzen l'algorisme *hash*. La funció més característica és *hash*. La funció *hash* genera una cadena utilitzant un algorisme i una font de dades.

La forma que té la funció *hash* és:

```
1 string **hash** (string %%%algorisme , string %%%data [, bool %%%  
    mode_binari= false ])
```

Els seus paràmetres indiquen el següent:

- *\$algorisme*: representa l'algorisme que s'ha d'utilitzar. Aquest algorisme pot ser *md5*, *sha1*, *gost*, *crc32*, *crc32b*, etc.
- *\$data*: són les dades que es filtraran per l'algorisme.
- *\$mode_binari*: és opcional i si el seu valor és *TRUE* marcarà que la cadena de sortida estigui formada per dades binàries; en canvi, si pren com a valor *FALSE* el resultat serà una cadena hexadecimal. Per defecte té el valor *FALSE*.

El codi següent:

```
1 <?php  
2 echo hash('md5', 'No oblideu la seguretat en els sistemes informàtics.',FALSE);  
3 ?>
```

crea la seqüència:

```
1 520b1e3205e719aa4d13a8c01b89e28d
```

En l'exemple anterior s'ha utilitzat l'algorisme MD5 per obtenir un codi *hash* hexadecimal. Per obtenir el codi és necessari tenir dades, que les podem obtenir d'una cadena de text. El resultat és en format hexadecimal i no en binari perquè hem indicat *FALSE* en el tercer paràmetre de la funció.

Normalment, s'utilitza una codificació *hash* per comprovar que el contingut d'un fitxer no s'ha modificat.

En aquest cas, la funció utilitzada és *hash_file*:

```
1 string **hash_file** ( string %%%algorisme , string %%%nom_fitxer [, bool  
    %%%mode_binari= false ] )
```

La funció *hash_file* retornarà una cadena. Aquesta és l'explicació dels seus paràmetres:

- *\$algorisme*: indica l'algorisme que s'ha d'utilitzar.
- *\$nom_fitxer*: indica l'arxiu que s'utilitzarà com a base per crear la codificació *hash*.
- *\$mode_binari*: és opcional, si el seu valor és *TRUE* la cadena resultant serà binària, i si el seu valor és *FALSE* el resultat es mostrarà en codificació hexadecimal.

Tiger és una funció *hash* pensada per a plataformes de 64 bits. Aquesta funció, entre d'altres, possiblement serà la que substituirà l'algorisme MD5.

Fixeu-vos en l'exemple següent, en què es calcula el codi *hash* d'un fitxer anomenat `exemple_hash_file.php`:

```
1 <?php
2 echo hash_file('md5', 'exemple_hash_file.php');
3 ?>
```

que dóna com a resultat:

```
1 2d205ca4c799ac669c97c93eadc5df2a
```

Podeu comprovar que si només utilitzeu els dos primers paràmetres la funció no dóna cap error i es mostra directament el resultat en codificació hexadecimal.

Després de fer una petita modificació en el fitxer `exemple_hash_file.php` el que es mostra en el navegador és:

```
1 7c804cc0743ef78c8cdaaa540ea332d2
```

Com podeu veure, el resultat és completament diferent. Per petita que sigui la modificació en el contingut d'un fitxer, el resultat d'aplicar un algoritme en una codificació *hash* implica obtenir una clau completament nova.

És molt important saber quin algoritme es pot utilitzar en l'ús d'aquestes funcions. El mateix llenguatge PHP us ofereix la possibilitat de consultar els algorismes suportats mitjançant la funció `hash_algos()`. Aquesta funció retorna una matriu que conté tots els algorismes que es poden utilitzar. A continuació, es mostra com es pot utilitzar aquesta funció:

```
1 <?php
2 print_r(hash_algos());
3 ?>
```

Generació i verificació de signatures en PHP

La seguretat en l'intercanvi d'informació a través de la Xarxa és una de les bases de la programació actual. El llenguatge PHP no podia oblidar implementar funcionalitats com ara la generació i verificació de signatures.

Mitjançant les funcions de l'OpenSSL, el llenguatge PHP pot generar certificats digitals i implementar protocols de comunicació segurs.

L'OpenSSL forma part d'un paquet. Abans de provar les funcions que veureu a continuació assegureu-vos que teniu instal·lat el paquet.

L'OpenSSL suporta els algorismes de signatura *DSS1*, *SHA1*, *MD5*, *MD4* i *MD2*. Per obtenir certificats heu de fer servir la funció *openssl_x509_read*:

```
1 resource openssl_x509_read ( mixed $x509certdata )
```

El paràmetre *\$x509certdata* és el certificat que es llegeix. Es retorna un identificador d'aquest certificat; aquest identificador es pot utilitzar en altres funcions PHP.

Per a la gestió de claus, les dues funcions principals són *open_get_publickey()* per obtenir una clau pública i *open_get_privatekey()* per obtenir una clau privada.

La descripció d'*open_get_publickey()* és un àlies de:

```
1 resource openssl_pkey_get_public ( mixed $certificat )
```

El paràmetre *\$certificat* pot ser un certificat X.509 o una clau privada.

La descripció d'*open_get_privatekey()* és un àlies de:

```
1 resource openssl_pkey_get_private ( mixed $clau [, string $passphrase= "" ] )
```

en què el paràmetre *\$clau* és la clau que es fa servir. Opcionalment, podeu utilitzar el paràmetre *\$passphrase* si la clau està encriptada.

X.509

X.509 és un estàndard que especifica, entre altres coses, certificats de clau pública. L'any 2007 es va demostrar que era vulnerable a determinats atacs.

PEM

PEM és un format de fitxer utilitzat per emmagatzemar xifrats digitals. Quan feu servir un paràmetre en una funció que sigui un certificat o una clau podreu fer servir fitxers d'aquest format.

2.3 Programació orientada a objectes en PHP

El llenguatge de programació PHP no és un llenguatge orientat a objectes, però sí que en suporta la gran majoria de característiques.

La programació orientada a objectes es fonamenta en dos pilars: les classes i els objectes. Amb aquests dos elements bàsics es poden dissenyar programes complexos molt potents d'una manera molt clara i senzilla.

Una **classe** en programació orientada a objectes és la definició d'un element que forma part d'un sistema.

Les classes defineixen la forma i el comportament dels objectes.

Els **objectes** són variables que responen a unes propietats i funcionalitats descrites en una classe.

Un exemple d'ús de la programació orientada a objectes seria:

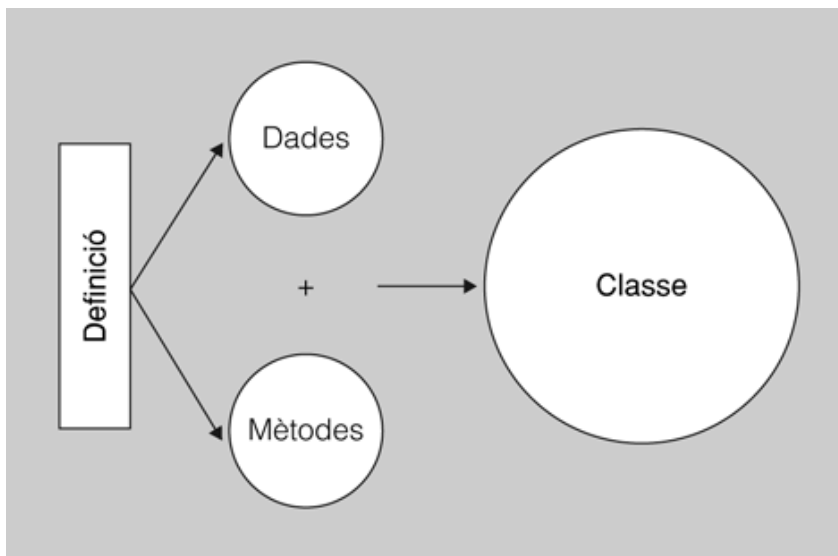
```
1 <HTML>
2   <head>
3     <title>Primer exemple amb P00</title>
4   </head>
5   <body>
6
7     <?php
8       class Alumne
9       {
10         var $nom;
11         var $dni;
12
13         public function anomenar($nom)
14         {
15           $this->nom=$nom;
16         }
17
18         public function dades($dni)
19         {
20           $this->dni=$dni;
21         }
22
23         public function imprimir()
24         {
25           echo $this->nom;
26           echo '<br>';
27           echo $this->dni;
28           echo '<br>';
29         }
30       }
31
32       $alumne1=new Alumne();
33       $alumne1->anomenar('Imena');
34       $alumne1->dades('23344556D');
35       $alumne1->imprimir();
36       $alumne2=new Alumne();
37       $alumne2->anomenar('Ketut');
38       $alumne2->dades('65544332E');
39       $alumne2->imprimir();
40     ?>
41   </body>
42 </HTML>
```

2.3.1 Les classes en PHP

Per definir una classe heu d'utilitzar la paraula reservada *class* seguida del nom de la classe. Heu de fer tot el possible perquè el nom de la classe sigui tan descriptiu com sigui possible respecte al que ha de representar. Entre claus heu d'escriure el codi de la classe.

```
1 class Alumne {}
```

En la figura 2.8 es representen les parts que defineixen una classe i com s'entrellacen entre elles. Podeu veure que les classes no són altra cosa que la definició de dades i mètodes, la unió dels quals genera la mateixa classe.

FIGURA 2.8. Definició de classe

Les classes poden tenir atributs i mètodes.

Els **atributs** són les variables que defineixen com són els objectes.

Per indicar els atributs d'una classe utilitzareu la cadena *var* davant del nom de l'atribut.

```
1 var $nom;  
2 var $dni;
```

Normalment, els atributs són privats, almenys així es recomana. D'aquesta manera, no s'hi pot accedir directament des de fora de la classe i s'ha de crear un mètode per accedir-hi. Per tant, quedaria:

```
1 private $nom;  
2 private $dni;
```

Els **mètodes** són les funcions que defineixen el comportament dels objectes.

Igual que els atributs, els mètodes també poden ser privats. Si us trobeu atributs o mètodes que no indiquen l'accessibilitat es consideren d'àmbit públic. Si voleu marcar explícitament que són públics heu de fer servir la partícula *public*.

En cas que es cridi un mètode dins el context d'un objecte, podreu utilitzar la pseudovariàble *\$this*, la qual fa referència al mateix objecte.

```
1 public function anomenar($nom)  
2 {  
3     $this->nom=$nom;  
4 }  
5  
6 public function dades($dni)  
7 {  
8     $this->dni=$dni;  
9 }
```

```
10
11 public function imprimir()
12 {
13     echo $this->nom;
14     echo '<br>';
15     echo $this->dni;
16     echo '<br>';
17 }
```

Vigileu de no confondre variables atributs i variables paràmetres. Fixeu-vos en l'exemple anterior en què apareix:

```
1 $this->dni=$dni;
```

\$this->dni fa referència a l'atribut d'un objecte i l'heu de considerar tot un grup; en canvi, *\$dni* és una variable paràmetre que arriba al mètode. No s'anomenen igual per confondre el programador, tot el contrari, resulta més senzill, ja que en definitiva en tots dos casos estem parlant del DNI, un és el DNI de l'alumne i l'altre és una variable que fa referència a un valor que s'acabarà assignant a un DNI.

2.3.2 Objectes en PHP

Per treballar amb classes cal que instancieu objectes. *Instanciar objectes* vol dir crear un ítem que és definit en la classe en què es referencia.

Els mètodes i les variables donaran personalitat als objectes que creeu.

Per instanciar un objecte caldrà que utilitzeu la paraula reservada *new*. Una vegada instanciats els objectes, penseu que cada un tindrà unes característiques pròpies (indicades en els atributs) i podrà fer una sèrie de funcions (definides en els mètodes).

A continuació, es mostra com s'instanciarien dos objectes de la classe Alumne:

```
1 $alumne1=new Alumne();
2 $alumne2=new Alumne();
```

Aquests objectes tindran els atributs i mètodes definits en la classe.

Si esteu gestionant la secretaria de l'escola i us arriben dos alumnes n'haureu de crear les fitxes. Per cada alumne es crearà un objecte mitjançant *new*. Però, com n'emmagatzemareu els noms? Així de senzill:

```
1 $alumne1->anomenar('Anna');
2 $alumne2->anomenar('Filomena');
```

La classe Alumne té un mètode que és anomenar, és a dir, donar nom. Per accedir als mètodes caldrà que utilitzeu *->* entre el nom de l'objecte i el mètode. Als mètodes, els podeu passar un paràmetre o més:


```
1 $alumne1->dades('23344556D');  
2 $alumne2->dades('65544332E');
```

Heu de tenir en compte que coincideixin el nombre de paràmetres d'entrada definits en la classe i el nombre de paràmetres que passeu al mètode.

El següent exemple, el mètode `dades` és definit en la classe `Alumne` i espera rebre tres paràmetres: un DNI, una data de naixement i un cicle formatiu.

```
1 <?php  
2  
3 class Alumne  
4 {  
5     var $dni;  
6     var $dataNaixement;  
7     var $cicleFormatiu;  
8  
9     public function dades($dni, $dataNaixement, $cicleFormatiu)  
10    {  
11        $this->dni = $dni;  
12        $this->dataNaixement = $dataNaixement;  
13        $this->cicleFormatiu = $cicleFormatiu;  
14    }  
15  
16    public function imprimir()  
17    {  
18        echo $this->dni . "<br>";  
19        echo $this->dataNaixement . "<br>";  
20        echo $this->cicleFormatiu . "<br>";  
21    }  
22  
23 }  
24  
25 $alumne1 = new Alumne();  
26 $alumne1->dades('12345678A', '12/12/12', 'ASIX');  
27 $alumne1->imprimir();  
28 $alumne2 = new Alumne();  
29 $alumne2->dades('98765432B', '1/1/1', 'DAM');  
30 $alumne2->imprimir();  
31  
32 ?>
```

Mètodes constructors

Normalment, totes les classes tenen un mètode que s'encarrega d'inicialitzar els objectes. No és usual anar creant els objectes per parts, sinó que hi ha un mètode que s'encarrega d'iniciar directament l'objecte.

Un **mètode constructor** és un mètode especial que s'encarrega d'inicialitzar els atributs de l'objecte que es crea; és el primer mètode que s'executa quan es crea un objecte i es crida automàticament.

L'exemple següent seria el mètode constructor d'una classe *Alumne*:

```
1 public function __construct($nom,$dni)  
2 {  
3     $this->nom=$nom;  
4     $this->dni=$dni;  
5 }
```

Per crear l'objecte caldrà que utilitzeu la partícula *new*:

```
1 $alumne3=new Alumne("Carles","39485764R");
```

Automàticament, el PHP va a buscar dins la classe *Alumne* el mètode constructor, que estarà sempre identificat amb la cadena *function __construct*.

Vigileu, no ho he escrivit malament! De vegades, segons el tipus de font no queda clar que abans de *construct* van dos guions baixos.

2.3.3 Herència en PHP

Per considerar un llenguatge com a orientat a objectes, aquest ha de tenir en compte l'herència obligatòriament.

L'**herència** permet crear noves classes a partir de classes ja existents; aquestes noves classes tenen els atributs i mètodes no protegits de les classes ja existents.

En el llenguatge de programació C++, l'herència múltiple és permesa, és a dir, que una classe hereti de més d'una classe.

En PHP l'herència és limitada, una classe només pot heretar d'una altra classe.

S'anomenen **superclasses** les classes de què deriven altres classes. Les classes **filles** són les que hereten atributs i mètodes.

Per utilitzar l'herència cal fer servir la paraula reservada *extends*.

A continuació, es mostra una classe anomenada *Persona*, que té una sèrie d'atributs i mètodes. Dues classes anomenades *Professor* i *Estudiant* hereten de *Persona*. Apareix una altra classe anomenada *Becari*, que hereta de la classe *Estudiant*. Podeu veure que hi podeu afegir més atributs i mètodes.

```
1 <html>
2 <head>
3   <title>Exemple amb Herència</title>
4 </head>
5 <body>
6
7 <?php
8
9   class Persona
10  {
11      private $nom;
12      private $dni;
13
14      public function __construct($nom,$dni)
15      {
16          $this->nom=$nom;
17          $this->dni=$dni;
18      }
19
20      public function dades_persona()
21      {
```

```
22     echo $this->nom.'<br>';
23     echo $this->dni.'<br><br>';
24 }
25 }
26
27 class Professor extends Persona
28 {
29     private $sou;
30     public function salari($sou)
31     {
32         $this->sou=$sou;
33     }
34 }
35
36 class Estudiant extends Persona
37 {
38     private $cicle;
39
40     public function matricular($cicle)
41     {
42         $this->cicle=$cicle;
43     }
44
45     public function dades_persona()
46     {
47         parent::dades_persona();
48         echo $this->cicle.'<br>';
49     }
50 }
51
52 class Becari extends Estudiant
53 {
54     private $sou;
55
56     public function salari($sou)
57     {
58         $this->sou=$sou;
59     }
60
61 }
62
63 $professorASI=new Professor('Lola','12345678A');
64 $professorASI->salari(10);
65 $alumneASI=new Estudiant('Joana','98765432S');
66 $alumneASI->matricular('ASI');
67 $becariASI=new Becari('Anna','38475619T');
68 $becariASI->salari(8);
69
70 echo $professorASI->dades_persona();
71 echo $alumneASI->dades_persona();
72 echo $becariASI->dades_persona();
73 ?>
74
75 </body>
76 </html>
```

Si observeu la classe Estudiant veureu la línia següent:

```
1 parent::dades_persona();
```

D'això, se'n diu *sobreescritura de mètodes*.

La **sobreescritura de mètodes** consisteix a aprofitar un mètode ja existent i redefinir-lo.

És possible que necessiteu ocultar dades a altres objectes, llavors utilitzareu *private*, però si treballeu amb herència podeu tenir problemes. Les classes filles no poden accedir als atributs i mètodes privats de les classes pare. Els llenguatges orientats a objectes ens ofereixen una solució: fer servir *protected*.

La classe i les subclasses poden accedir a un atribut o un mètode **protected**, però no els objectes creats a partir d'aquestes classes.

En l'exemple següent es protegeix l'atribut *\$sou*. Per poder-lo canviar s'ha creat la funció *salari*:

```
1 <html>
2 <head>
3   <title>Exemple amb protected</title>
4 </head>
5 <body>
6
7 <?php
8   class Persona
9   {
10     var $nom;
11     private $dni;
12     protected $sou;
13
14     public function anomena($nom)
15     {
16         $this->nom=$nom;
17     }
18
19     public function fitxa($dni)
20     {
21         $this->dni=$dni;
22     }
23
24     public function salari($sou)
25     {
26         $this->sou=$sou;
27     }
28
29     public function mostra_dades()
30     {
31         echo $this->nom.'<br>';
32         echo $this->dni.'<br>';
33         echo $this->sou.'<br>';
34     }
35 }
36
37
38 class Professor extends Persona
39 {
40     private $especialitat;
41
42     public function assigna_especialitat($especialitat)
43     {
44         $this->especialitat=$especialitat;
45     }
46 }
47
48 $nouProfessor=new Professor();
49 $nouProfessor->nom='Andrea';
50 $nouProfessor->dni='38293847Y';
51 // $nouProfessor->sou=10;
52 $nouProfessor->salari(10);
53 $nouProfessor->assigna_especialitat('Informàtica');
54 echo $nouProfessor->mostra_dades();
```

```
55 ?>
56 </body>
57 </html>
```

Si descomenteu la línia,

```
1 // $nouProfessor->sou=10;
```

us donarà l'error següent:

```
1 Fatal error: Cannot access protected property Professor::$sou
```

És a dir, esteu intentant accedir directament a un valor protegit. Aprofitant l'exemple anterior podreu veure que s'està intentant donar valor a l'atribut *dni*, quan aquest és privat; això no genera error, però no fa res, no emmagatzema cap valor.

3. Accés a bases de dades des de PHP

El major benefici de treballar amb pàgines dinàmiques és que podeu agafar la informació que s'hi mostrarà d'una base de dades, en lloc d'haver de tenir-la dins del codi o a un fitxer extern. L'ús de MySQL amb PHP és molt popular en el desenvolupament d'aplicacions web.

3.1 Gestors de bases de dades més usats

Existeixen diferents bases de dades al mercat que podeu fer servir per crear la vostra pàgina web amb bases de dades. És important conèixer-ne les més populars per tal de poder decidir en cada cas quina serà l'òptima per vosaltres.

- **MySQL:** és una plataforma gratuïta i molt adequada per aprendre. És possiblement una de les bases de dades més ràpides que podem trobar, a més de consumir pocs recursos de la màquina on està instal·lada. Es tracta d'un sistema relativament fàcil d'instal·lar i administrar enfront d'altres productes del mercat. Es recomana per iniciar-se en el món de les bases de dades, ja que disposa de moltes utilitats, manuals i documentació que la immensa comunitat d'usuaris s'ha encarregat de realitzar desinteressadament. La gestió de la base de dades utilitza l'SQL (*Structured Query Language*), a més d'utilitzar protocols de comunicació de bases de dades desenvolupades per Microsoft, mitjançant ODBC o ADO.
- **PostgreSQL:** és un gestor de bases de dades usat principalment per grans organitzacions en aplicacions crítiques o importants. PostgreSQL és la segona base de dades més popular implantada en el mercat GNU/Linux. El PostgreSQL es va dissenyar com una base de dades orientada a objectes, és a dir, una ORDBMS, amb la qual el concepte de *taules* el substituïm per *objectes*, i es permet crear nous tipus de dades, fer herències entre objectes, etc. El PostgreSQL és, sens dubte, una de les bases de dades que ha aportat més implementacions per als experts. Disposa d'allò que al MySQL li falta, però també li falta tot allò que el MySQL té. El PostgreSQL té transaccions, integritat referencial, vistes i un gran nombre de funcionalitats, a canvi de convertir-se en un motor més lent i pesat que MySQL.
- **Oracle:** és un dels sistemes de gestió de bases de dades relacional més importants. Està disponible en una varietat de configuracions, des de petites versions personals fins a versions empresarials.
- **Microsoft Access:** és un sistema de pagament que permet fer un disseny de la base de dades mitjançant una eina gràfica. Té certes limitacions

ORDBMS
(*Object-Relational Database Management System*, sistema de gestió de bases de dades objecte-relacionals) és un sistema gestor de bases de dades similar als relacionals, però amb un model de bases de dades objecte-relacionals amb objectes, classes i herència.

importants (com ara el número de connexions concurrents que pot tractar) i hi ha pocs sistemes operatius sobre els quals pot funcionar.

- Microsoft SQL Server: està dissenyat per crear webs i sistemes de bases de dades empresarials d'escriptori. Permet emmagatzemar grans quantitats d'informació comparat amb l'Access.

Farem servir MySQL perquè: és ideal per aplicacions petites, encara que al ser escalable també es pot fer servir per aplicacions grans; és molt ràpid, fiable i fàcil d'usar; fa servir SQL estàndard; compila una gran quantitat de plataformes; i, finalment, perquè és gratuït de descarregar i usar.

3.2 Gestió de bases de dades amb PHPMyAdmin

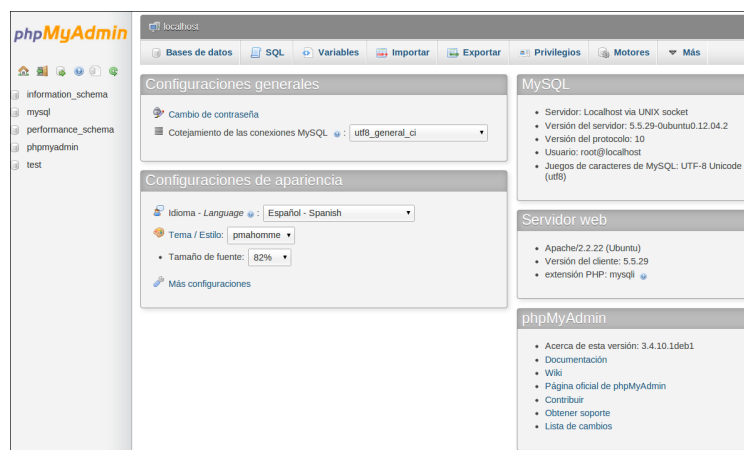
PHPMyAdmin és una eina lliure i gratuïta escrita en PHP que permet controlar l'administració de MySQL des d'un navegador web. Permet realitzar diferents tasques com ara crear, modificar o esborrar bases de dades, taules, camps o files... executant ordres SQL.

Si s'hi accedeix amb un usuari que sigui administrador del MySQL, el phpMyAdmin permet gestionar totes les bases de dades existents del servidor. Altrament, només es mostren les bases de dades a les quals es té accés. Els usuaris administradors tenen accés a una base de dades anomenada *mysql* (base de dades de control) que serveix per configurar el motor de bases de dades. És molt important no tocar-la ni fer-hi canvis a no ser que es sàpiga perfectament el que s'està fent. En cas contrari, el MySQL podria deixar de funcionar o quedar inoperatiu.

Per accedir-hi heu d'entrar des del navegador a l'adreça <http://localhost/phpmyadmin>.

Una vegada introduïdes les dades d'accés (nom i contrasenya) heu de veure una pantalla com la de la figura 3.1.

FIGURA 3.1. Pantalla inicial de PHPMyAdmin



El phpMyAdmin divideix la finestra en dos marcs. El de l'esquerra ofereix un menú desplegable amb les bases de dades a les quals es té accés (si són més d'una) i les taules de la base de dades seleccionada. El marc de la dreta és el lloc en el qual es mostra la informació de la navegació. Inicialment mostra una pantalla de benvinguda amb diversos enllaços.

En clicar al nom de les taules que es mostren en el marc de l'esquerra es pot veure'n l'estructura (els camps) en el marc de la dreta i apareix un menú horitzontal que permet accedir a les diverses funcionalitats de l'aplicació.

Si seleccioneu una base de dades existent (per exemple *mysql*) veureu per defecte un llistat de totes les taules que existeixen en la base de dades, tal com podeu veure a la figura 3.2.

FIGURA 3.2. Vista de base de dades

Tabla	Acción	Filas	Tipo	Cotejamiento	Tamaño	Residuo a depurar
columns_priv	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_bin	4.0 KB	-
db	Examinar Estructura Buscar Insertar Vaciar Eliminar	3	MyISAM	utf8_bin	6.3 KB	-
event	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	2.0 KB	-
func	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_bin	1.0 KB	-
general_log	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	CSV	utf8_general_ci	unknown	-
help_category	Examinar Estructura Buscar Insertar Vaciar Eliminar	39	MyISAM	utf8_general_ci	25.1 KB	-
help_keyword	Examinar Estructura Buscar Insertar Vaciar Eliminar	464	MyISAM	utf8_general_ci	105.3 KB	-
help_relation	Examinar Estructura Buscar Insertar Vaciar Eliminar	1,028	MyISAM	utf8_general_ci	28.0 KB	-
help_topic	Examinar Estructura Buscar Insertar Vaciar Eliminar	508	MyISAM	utf8_general_ci	455.9 KB	-
host	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_bin	2.0 KB	-
ndb_binlog_index	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	latin1_swedish_ci	1.0 KB	-
plugin	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	1.0 KB	-
proc	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	4.3 KB	320 B
proc_priv	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_bin	4.0 KB	-
proxies_priv	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	MyISAM	utf8_bin	6.4 KB	-
servers	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	1.0 KB	-
slow_log	Examinar Estructura Buscar Insertar Vaciar Eliminar	2	CSV	utf8_general_ci	unknown	-
tables_priv	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_bin	4.0 KB	-
time_zone	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	1.0 KB	-
time_zone_leap_second	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	1.0 KB	-
time_zone_name	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	1.0 KB	-
time_zone_transition	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	1.0 KB	-
time_zone_transition_type	Examinar Estructura Buscar Insertar Vaciar Eliminar	0	MyISAM	utf8_general_ci	1.0 KB	-
user	Examinar Estructura Buscar Insertar Vaciar Eliminar	8	MyISAM	utf8_bin	2.7 KB	-
24 tablas	Número de filas	2,056	InnoDB	latin1_swedish_ci	659.0 KB	320 B

Si feu clic a qualsevol de les taules, podreu veure l'estructura de la taula, examinar-ne el registres o inserir-ne un de nou.

Fent clic a la pestanya SQL podeu inserir consultes directament fent servir sintaxi SQL. Això és molt útil quan esteu programant, ja que podeu comprovar si les consultes que esteu realitzant durant el programa són correctes o no. Simplement copieu el text de la consulta a la caixa i feu clic a *Continuar*. Podreu veure els registres resultants de la consulta que heu fet.

Finalment, amb l'opció *Exportar* podeu extraure les dades de la base de dades a un fitxer de text pla que podreu importar posteriorment, amb l'opció *Importar*, a una base de dades MySQL.

3.3 Connexió amb el servidor MySQL

L'estructura per accedir a una base de dades és molt semblant al protocol per accedir a un fitxer:

1. Obrir la base de dades.
2. Trametre la comanda SQL a la base de dades.
3. Retornar el resultat de la consulta.
4. Tancar la connexió de la base de dades.

Abans de poder accedir a les dades en una base de dades, s'ha de crear una connexió amb la base de dades. En PHP per connectar amb la base de dades MySQL això es fa amb la funció `mysql_connect()`.

```
1 recurs = mysql_connect(nom del servidor, nom d'usuari, contrasenya, nou enllaç,  
senyals de client);
```

La funció `mysql_connect()` retorna una connexió amb el servidor MySQL Server en cas d'èxit, o `FALSE` i un error en cas de fallada. Podeu amagar la sortida d'error mitjançant l'addició d'una `@` davant del nom de la funció.

Aquesta és la descripció dels paràmetres:

- Nom del servidor: és opcional. Especifica el servidor MySQL al qual connectar. Pot incloure un número de port, per exemple *Servidor: port* o una ruta, per exemple, *:/camí/al/directori* per al *localhost*. Si la directiva de PHP *mysql.default_host* no està definida (per defecte), llavors el valor per defecte és *localhost: 3306*. En el mode SQL segur, aquest paràmetre s'ignora i *localhost: 3306* s'utilitza sempre.
- Nom d'usuari: és opcional. Especifica el nom d'usuari per iniciar la sessió. El valor per defecte és el nom de l'usuari propietari del procés del servidor que està definit per la directiva *mysql.default_user*. En el mode SQL segur, aquest paràmetre s'ignora i s'usa el nom de l'usuari propietari del procés del servidor.
- Contrasenya: és opcional. Especifica la contrasenya per iniciar sessió. El valor per defecte és definit per la directiva *mysql.default_password*. En el mode SQL segur, aquest paràmetre no es té en compte i s'usa la contrasenya buida.
- Nou enllaç: és opcional. Si es fa una segona trucada a `mysql_connect()` amb els mateixos arguments, no s'estableix un nou enllaç, en el seu lloc, es retornarà l'identificador d'enllaç de l'enllaç ja obert. El paràmetre nou enllaç modifica aquest comportament i fa que `mysql_connect()` sempre obri un nou vincle, encara que s'hagi cridat a `mysql_connect()`, amb els mateixos paràmetres. En el mode SQL segur, aquest paràmetre s'ignora.

- Senyals de client: és opcional. El paràmetre senyals de client pot ser una combinació de les següents constants: 128 (habilita el control de LOAD DATA LOCAL), MYSQL_CLIENT_SSL, MYSQL_CLIENT_COMPRESS, MYSQL_CLIENT_IGNORE_SPACE o MYSQL_CLIENT_INTERACTIVE. En el mode SQL segur, aquest paràmetre s'ignora.

En el següent exemple s'emmagatzema la connexió en una variable (\$connexio) per al seu ús posterior. Si la connexió falla s'executarà el codi de dins de l'*if* i el programa sortirà.

```
1 <?php
2
3 $connexio = mysql_connect("localhost", "admin", "12345");
4     if (!$connexio)
5     {
6         exit('No es pot connectar:'. mysql_error());
7     }
8
9     echo "Connexió correcta!";
10 ?>
```

La funció `exit()` mostra un missatge i surt de l'execució del codi. La funció `mysql_error()` retorna un text amb un missatge d'error de la darrera operació MySQL que s'ha executat.

3.4 Tancar la connexió a la base de dades

La funció de PHP `mysql_close()`, tanca una connexió MySQL. Aquesta funció retorna TRUE si té èxit, o FALSE en cas de fallada.

Té un únic argument opcional que és la connexió MySQL per tancar. Si no s'especifica, s'utilitza l'última connexió oberta per `mysql_connect()`.

L'ús de `mysql_close()` no és imprescindible, ja que les connexions no persistents són tancades automàticament al final de l'execució de l'script PHP.

Aquí teniu un exemple d'utilització:

```
1 <?php
2
3 $connexio = mysql_connect("localhost", "admin", "12345");
4     if (!$connexio)
5     {
6         exit('No es pot connectar:'. mysql_error());
7     }
8
9     echo "Connexió correcta!<br>";
10
11     mysql_close($connexio);
12
13     echo "Connexió tancada";
14 ?>
```

3.5 Selecció de la base de dades

Abans de començar a treballar amb una base de dades del servidor, l'hem de seleccionar. Per fer-ho, existeix la funció `mysql_select_db($nom, $connexio)` que agafa dos arguments que especifiquen el nom de la base de dades que volem escollir, i la connexió al servidor que hem fet amb anterioritat, respectivament. La funció retorna `TRUE` en cas d'èxit, i `FALSE` en cas d'error.

Aquí teniu un exemple d'utilització de la funció:

```
1 $connexio = mysql_connect("localhost", "usuari", "12345");
2
3     if (!$connexio)
4     {
5         exit('No es pot connectar:'. mysql_error());
6     }
7
8     if(!mysql_select_db("taula_equips", $connexio))
9         exit("Error al connectar amb la base de dades". mysql_error());
```

3.6 Treball amb la base de dades

Amb una base de dades es poden fer diverses operacions: consultar dades, inserir dades, actualitzar dades i esborrar dades.

Per aquestes operacions l'estructura del codi de programació és exactament igual. Sempre que es programa una connexió a una base de dades MySQL s'han de fer els passos següents:

1. Connexió al servidor MySQL: per fer aquesta operació s'utilitza la funció `mysql_connect($servidor, $usuari, $contrasenya)`. La funció retorna un valor del tipus *resource* (recurs), que representa la connexió amb la base de dades.
2. Selecció de la base de dades amb la qual es treballa: amb `mysql_select_db($nom_BD, $connexió)`. S'ha d'indicar el nom de la base de dades amb la qual es vol treballar i de la connexió amb el servidor.
3. Fer la consulta/operació necessària amb la funció `mysql_query($consulta)`.
4. Podeu comprovar quantes files ha retornat un `SELECT` o han estat inserides modificades. Per consultar quantes files ha retornat un consulta, podeu fer servir la funció `mysql_num_rows ($result)` que retorna el número de files que s'han seleccionat amb un `SELECT` o `SHOW`. Per consultar quantes files han estat inserides, modificades o esborrades amb `INSERT`, `UPDATE`, `REPLACE` o `DELETE`, podeu fer servir la funció `mysql_affected_rows()`.

5. Si és una operació de consulta de dades (com un SELECT), falta llegir les dades que retorna la consulta SQL. Per fer-ho podeu fer servir la funció `mysql_fetch_array($result)` que retorna els valors en forma de matriu associativa en la qual les claus són les columnes de la taula. Retorna FALSE si no hi ha més files.

3.6.1 Inserció d'informació

Per afegir registres a la base de dades heu de fer servir una consulta MySQL de tipus INSERT que té la següent estructura:

```
1 INSERT INTO nom_taula [(columna1, columna2, ...)]
2     VALUES (valor1, valor2,...);
```

Podeu consultar la sintaxi completa a:

<http://dev.mysql.com/doc/refman/5.5/en/insert.html>

Per exemple:

```
1 "INSERT INTO discos (titol, autor, any) VALUES ('Axis: Bold as love', 'The Jimi
   Hendrix Experience', '1967');"
```

El següent exemple mostra un codi sencer d'inserció de dades a una base de dades:

```
1 $usuari = "admin";
2 $contrasenya = "admin";
3
4
5
6 // Connexió amb el servidor de base de dades i selecció de la base de dades
7 $connexio = mysql_connect("localhost", $usuari, $contrasenya);
8
9 if (!$connexio)
10     exit('No es pot connectar:'. mysql_error());
11
12 if(!mysql_select_db("llibreria", $connexio))
13     exit("Error al connectar amb la base de dades". mysql_error());
14
15 //Com que no inserirem totes les dades (id és autoincremental) s'han d'
   especificar els camps
16 $consulta = "INSERT INTO llibres (titol, autor, any) VALUES ('Norwegian Wood',
   'Haruki Murakami', '2000')";
17
18 // Fem la consulta al servidor i obtenim la resposta
19 $resultat = mysql_query($consulta, $connexio);
20
21
22 if (!$resultat)
23 {
24     $missatge = 'Query incorrecta: ' . mysql_error() . "\n";
25     $missatge .= 'Query sencera: ' . $consulta;
26     exit($missatge);
27 }
28
29
30 echo "Dades introduïdes correctament!!";
```

Podeu comprovar posteriorment que s'han inserit les dades (amb una consulta MySQL o des de PHPMyAdmin, per exemple).

La gràcia de les pàgines dinàmiques és que la informació que s'ha d'inserir a la base de dades no és estàtica, com en aquest exemple, sinó que són valors que introdueixen els usuaris. Aquesta tasca es fa mitjançant l'ús de formularis. La seqüència d'accions és la següent:

1. L'usuari introdueix dades amb un formulari.
2. Quan valida el formulari amb el botó, s'envien les dades utilitzant el mètode *POST* o *GET*.
3. Rebudes les dades del formulari es fa la connexió a la base de dades i la inserció de les dades.

3.6.2 Consulta d'informació

Per realitzar una consulta a la base de dades per obtenir informació, heu de fer servir una sentència *SELECT*, que té la següent forma:

```
1 SELECT  
2   expressió  
3   FROM taula  
4   WHERE condició  
5   [GROUP BY {columna | expressió | posició}  
6     [ASC | DESC]]  
7   [HAVING condició]  
8   [ORDER BY {columna | expressió | posició}  
9     [ASC | DESC]]  
10  [LIMIT {[desplaçament,] num_files | num_files OFFSET offset}]
```

Podeu consultar la sintaxi completa a:

<http://dev.mysql.com/doc/refman/5.6/en/select.html>

Per exemple:

```
1 SELECT nom, dni FROM usuaris WHERE uid=1
```

Una vegada creada la consulta i emmagatzemada a una variable, podeu executar-la sobre la base de dades amb la funció `mysql_query($consulta, $connexio)`.

www.php.net/manual/en/function.mysql-query.php

Aquesta funció:

- Per consultes del tipus *SELECT*, *SHOW*, *DESCRIBE*, *EXPLAIN* retorna una variable de tipus *resource* (recurs) en cas d'èxit i *FALSE* en cas d'error.
- Per consultes *INSERT*, *UPDATE*, *DELETE*, *DROP*, retorna *TRUE* en cas d'èxit i *FALSE* en cas d'error.

Una vegada heu fet la consulta SELECT, podeu esbrinar quantes files ha retornat la consulta amb la funció `int mysql_num_rows($result)` que retorna el número de files que ha retornat el SELECT o FALSE en cas d'error.

Per obtenir les dades del resultat de la consulta feu servir la funció `array mysql_fetch_array ($resultat, $tipus)` que retorna un *array* associatiu que correspon a una fila del resultat de la consulta (i avança el punter que indica la fila a mostrar).

L'argument `$tipus` indica el tipus de l'*array* associatiu que s'obté. És una constant que pot agafar els següents valors: `MYSQL_ASSOC`, `MYSQL_NUM` o `MYSQL_BOTH` (que és el valor per defecte). Si deixeu aquest paràmetre en blanc o seleccioneu `MYSQL_BOTH` obtindreu un *array* amb el qual podeu accedir als elements tant pel número de la columna, com pel seu nom.

La funció va retornant les diferent files resultants del SELECT segons com és cridada, fins que arriba a l'última fila resultant, en la qual retorna FALSE. Per tant, per obtenir totes les dades de retorn d'un SELECT, heu de cridar a `mysql_fetch_array` dins d'un bucle fins que la funció us retorni FALSE. Per exemple:

```
1 $result = mysql_query("SELECT id, nom FROM taula");
2
3 while ($fila = mysql_fetch_array($result, MYSQL_BOTH))
4 {
5     printf("ID: %s  Nom: %s", $fila[0], $fila[1]);
6
7     //com fem servir MYSQL_BOTH, és igual a
8     printf("ID: %s  Nom: %s", $fila["id"], $fila["nom"]);
9 }
```

En aquest altre exemple es mostren els continguts dins d'una taula.

```
1 // Connexió amb el servidor de base de dades i selecció de la base de dades
2 $connexio = mysql_connect("localhost", $usuari, $contrasenya);
3
4 if (!$connexio)
5     exit('No es pot connectar:'. mysql_error());
6
7 if(!mysql_select_db("llibreria", $connexio))
8     exit("Error al connectar amb la base de dades". mysql_error());
9
10 //Definim la consulta que recuperarà totes les dades de la taula llibres
11 $consulta="SELECT * FROM llibres";
12
13 //Fem la consulta al servidor i obtenim la resposta
14 $resultat = mysql_query( $consulta, $connexio);
15
16 //Si tenim resultats els mostrarem
17 if( mysql_num_rows($resultat) > 0)
18 {
19
20     //Creem una taula per posar els resultats
21     echo "<table border = '1'>";
22     echo "<tr>";
23     echo "<td>id</td>";
24     echo "<td>titol</td>";
25     echo "<td>autor</td>";
26     echo "<td>editorial</td>";
27     echo "</tr>";
28
29     // Mentre hi hagi resultats els afegim a la taula
```

```
30 while($fila = mysql_fetch_array($resultat))
31 {
32     //Hi afegim una filera i recuperem els valors utilitzant la matriu
33     associativa que guardem a $fila a cada iteració
34     echo "<tr>";
35     echo "<td>".$fila['id']. "</td>";
36     echo "<td>".$fila['titol']. "</td>";
37     echo "<td>".$fila['autor']. "</td>";
38     echo "<td>".$fila['editorial']. "</td>";
39     echo "</tr>";
40 }
41
42 //fi de la taula
43 echo "</table>";
44 }
45 else
46 {
47     echo "No hi ha cap resultat";
48 }
```

3.6.3 Modificació d'informació

Per modificar la informació de la base de dades heu de fer servir la consulta UPDATE de MySQL que té la següent forma:

```
1 UPDATE taula
2     SET nomColumna1={expressió1|DEFAULT},
3       nomColumna2={expressió2|DEFAULT}...
4
5     WHERE condició
6     [LIMIT numero_files]
```

Podeu veure'n la sintaxi completa a:

<http://dev.mysql.com/doc/refman/5.6/en/update.html>

Per exemple:

```
1 UPDATE taula SET id='200' WHERE id=2
```

O

```
1 UPDATE taula1 SET col1 = col1 + 1, col2 = col1;
```

No cal posar les cometes a la condició WHERE.

La fareu servir mitjançant la funció `mysql_query($consulta)`. Posteriorment podeu fer servir la funció `mysql_affected_rows()` per obtenir la quantitat de files modificades amb la sentència.

En el següent exemple es modifiquen les dades a una base de dades en funció dels valors enviats per un formulari HTML.

```
1 //Connexió amb la base de dades i selecció base dades llibreria
2 $enllac = connexioBaseDades();
```



```
3
4 if ( isset($_POST['actualitza']) )
5 {
6     //L'usuari ha decidit enviar al servidor les noves dades actualitzades
7     $consulta = "UPDATE llibres SET titol='".$_POST['titol']."', autor='".$_
        $_POST['autor']. "', editorial='".$_POST['editorial']."' WHERE id="
        $_POST['id'];
8
9     $resultat = mysql_query( $consulta, $enllac);
10
11     $err = mysql_error();
12     if( $err != "" )
13         echo "error=$err <br>";
14
15     printf("Registres modificats: %d\n", mysql_affected_rows());
16
17
18 }
```

3.6.4 Esborrar registres

Per esborrar registres d'una taula heu de fer servir una consulta DELETE que té la següent forma:

```
1 DELETE FROM taula
2     [WHERE condició]
3     [ORDER BY ...]
4     [LIMIT numero_files]
```

Podeu trobar-ne la sintaxi sencera a:

<http://dev.mysql.com/doc/refman/5.6/en/delete.html>

Per exemple:

```
1 DELETE FROM taula WHERE id < 10
```

La fareu servir mitjançant la funció `mysql_query($consulta)`. Posteriorment podeu fer servir la funció `mysql_affected_rows()` per obtenir la quantitat de files modificades amb la sentència.

Al següent exemple s'esborra un element especificat per un formulari HTML de la taula *llibres*:

```
1 if( isset($_POST['id']) )
2 {
3     //Definim la consulta que esborra el registre seleccionat
4     $consulta="DELETE FROM llibres WHERE id=".$_POST['id'];
5     //Fem la consulta al servidor i obtenim la resposta
6     $resultat = mysql_query($consulta, $connexio);
7
8     //Mireu una forma alternativa de veure si hi ha hagut un error
9     $err = mysql_error();
10     if( $err != "" )
11         echo "error=$err <br>";
12
13     printf("Registres esborrats: %d\n", mysql_affected_rows());
14 }
```

3.6.5 Gestió d'errors

La funció `mysql_error($connexio)` retorna un text amb la descripció de l'error de l'última funció MySQL que s'ha executat. Per tant, l'heu d'executar justament després de la funció de la qual voleu saber l'error.

En el cas de que no hi hagi cap error, retorna una cadena buida (`""`).

El paràmetre `connexió` és opcional. Especifica la connexió MySQL. Si no s'especifica, s'utilitza l'última connexió oberta per `mysql_connect()`.

La funció `mysql_errno` retorna el número d'error de la sentència SQL anterior.

En aquest exemple podeu veure com tractar una sèrie d'errors treballant amb bases de dades:

```
1 <?php
2 $connexio = mysql_connect("localhost", "usuari", "contrasenya");
3
4 mysql_select_db("bd_inexistent", $connexio);
5 echo mysql_errno($connexio) . ": " . mysql_error($connexio) . "\n";
6
7
8 mysql_select_db("coses", $connexio);
9 mysql_query("SELECT * FROM bd_inexistent", $connexio);
10 echo mysql_errno($connexio) . ": " . mysql_error($connexio) . "\n";
11 ?>
```

3.7 Creació d'una base de dades

Hi ha diverses formes de crear una base de dades. Generalment la creareu amb una eina com pot ser PHPMyAdmin, encara que ho podeu fer també des del codi.

Per crear una base de dades a MySQL es fa servir la sentència:

```
1 mysql> CREATE DATABASE Nom_base_de_dades;
```

En ambients Unix, els noms de les bases de dades són *case sensitive* (al contrari que les paraules clau), de manera que sempre s'ha de referir a la base de dades exactament igual (i no `nom_base_de_dades` o `NOM_BASE_DE_DADES`). Això també s'aplica als noms de taules. Aquesta restricció no existeix a Windows, encara que pot utilitzar el mateix esquema de majúscules quan es refereixi a bases de dades i taules en una consulta donada.

La comanda `mysql_query()` de PHP envia una sentència a la base de dades. Per tant podeu fer-la servir per crear una base de dades al servidor:

```
1 <?php
2 $sql = 'CREATE DATABASE LaMevaBD';
3
4 // $con conté la connexió a la BD
```

```
5 if (mysql_query($sql, $con))
6 {
7     echo "S'ha creat la Base de dades LaMevaBD\n";
8 } else {
9     echo 'Error al crear la base de dades: ' . mysql_error() . "\n";
10 }
11 ?>
```

En crear una base de dades, aquesta no es selecciona per al seu ús automàticament, s'ha de fer explícitament. La sintaxi per fer-ho a MySQL és

```
1 mysql> USE Nom_base_de_dades
```

Les bases de dades només necessiten ser creades un sol cop, però han de ser seleccionades cada vegada que s'inicia una sessió de MySQL. Es pot fer a través de la comanda **USE** com es mostra a l'exemple. Quan accediu al MySQL des de consola ho podeu indicar en la línia de comandes en executar MySQL. Simplement s'ha d'indicar el nom de la base de dades a continuació dels paràmetres que necessiteu per ingressar. Per exemple:

```
1 shell> mysql -h host -o user -p LaMevaBD
```

Advertiu que en la comanda anterior LaMevaBD no és la contrasenya. Si es volgués subministrar la contrasenya en la línia de comandes, després de l'opció -p, s'ha de fer sense deixar espais en blanc (per exemple, -pcontrasenya, no -p contrasenya). De tota manera, escriure la contrasenya en la línia de comandes no és recomanable perquè ho exposa a la vista d'altres usuaris.

Per eliminar una base de dades a PHP, igual que per la creació d'una base de dades, utilitzareu la comanda `mysql_query()` per executar la instrucció **DROP DATABASE**, tal com es pot veure en el següent exemple:

```
1 <?php
2 $query = 'DROP DATABASE LaMevaBD';
3 $result = mysql_query($query,$con);
4 ?>
```

Podeu veure el contingut de les taules amb l'ordre de MySQL **SHOW TABLES** directament sobre MySQL:

```
1 mysql> SHOW TABLES;
2 Empty set (0.00 sec)
```

En un principi la base de dades recent creada està buida.

3.7.1 Creació de taules

A MySQL podeu emprar la sentència **CREATE TABLE** per especificar l'estructura d'una taula:

```

1 CREATE TABLE nom_taula
2 (
3     nom_columna1 tipus_dades,
4     nom_columna2 tipus_dades,
5     nom_columna3 tipus_dades,
6 )

```

Sempre que voleu executar ordres al MySQL podeu fer servir la funció de PHP `mysql_query()`. En aquest cas s'ha d'executar la instrucció de MySQL `CREATE TABLE`, com es pot veure en el següent exemple:

```

1 <?php
2 // Seleccioneu la Base de Dades
3 mysql_select_db("LaMevaBD", $con);
4 $sql = "CREATE TABLE nom_taula
5 (
6     nom varchar(15),
7     cognom varchar(15),
8     anys int
9 );";
10
11 // Executem la consulta
12 mysql_query($sql,$con);
13 ?>

```

Després de crear la taula, des de MySQL amb la comanda `SHOW TABLES` s'hauria de produir la següent sortida:

```

1 mysql> SHOW TABLES;
2 +-----+
3 | Tables in LaMevaBD |
4 +-----+
5 | nom_taula          |
6 +-----+

```

Per verificar que la taula ha estat creada en la forma esperada, utilitzeu la sentència `DESCRIBE` a MySQL:

```

1 mysql> DESCRIBE nom_taula;
2 +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
3 | Field | Type  | Null  | Key  | Default | Extra | |
4 +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+
5 | col1  | tipus | ca    |      | NULL    |       | |
6 | col2  | tipus | ca    |      | NULL    |       | |
7 | col3  | tipus | ca    |      | NULL    |       | |
8 +-----+ +-----+ +-----+ +-----+ +-----+ +-----+ +-----+

```

La sentència `DESCRIBE` pot ser utilitzada en qualsevol moment, per exemple, si oblideu els noms o el tipus de dades de les columnes de la taula.

Per acabar veureu un exemple on es crea la base de dades *Alumnes_IOC* i s'afegeix una taula *alumnes* amb tres columnes *nom*, *cognom* i *anys*.

```

1 <?php
2 $dbhost = 'localhost';
3 $dbuari = 'root';
4 $dbcon = 'contrasenya';
5 $con = mysql_connect($dbhost, $dbuari,$dbcon);
6
7 if (!$con)
8 {

```

```
9      exit('Error al connectar amb la base de dades de mysql: ' . mysql_error()
10    );
11  }
12  $sql = 'CREATE DATABASE Alumnes_IOC';
13
14  if (mysql_query($sql, $con))
15  {
16      echo "S'ha creat la Base de dades Alumnes_IOC\n";
17  }
18  else
19  {
20      echo 'Error al crear la base de dades: ' . mysql_error() . "\n";
21  }
22
23  // Seleccioneu la Base de Dades
24  mysql_select_db("Base_de_Dades", $con);
25  $sql = "CREATE TABLE alumnes
26  (
27      nom varchar(15),
28      cognom varchar(15),
29      anys int
30  ) ";
31
32  // Executau la consulta
33  mysql_query($sql,$con);
34
35  // Tancau la connexió
36  mysql_close($con);
37
38  ?>
```

3.8 Mecanismes de seguretat

La seguretat és un tema complex que requereix una planificació a tots els nivells en què s'està treballant la pàgina web i la base de dades. Per una banda fareu servir contrasenyes que el més adequat és que guardeu de forma encriptada a la base de dades.

Els usuaris únicament han de poder accedir a les pàgines del web a les quals tenen permisos. Per exemple, un usuari alumne de l'IOC no té accés a l'edició de l'aula.

Per altra banda, els usuaris de la base de dades, han de tenir únicament la capacitat d'accedir a les dades estrictament necessàries. Els usuaris dins de la base de dades també tenen un accés limitat a ella.

3.8.1 Codificació de la informació

Podeu fer servir l'algorisme MD5 per codificar les dades confidencials dins de la base de dades. Podeu trobar-ne l'especificació aquí:

www.faqs.org/rfcs/rfc1321.html

Afortunadament MD5 està implementat directament a PHP, per això podeu calcular l'MD5 d'una cadena simplement cridant a una funció.

La funció `string md5 ($cadena)` encripta la cadena fent servir l'algorisme MD5 i la retorna en forma de número hexadecimal de 32 caràcters.

L'ús de l'algorisme SHA1 o MD5 no és estrictament segur per emmagatzemar les dades, tal com podeu veure a aquest article de PHP.net (www.php.net/manual/en/faq.passwords.php#faq.passwords.fasthash), però per fer les nostres proves tindrà una seguretat suficient i servirà per aprendre a treballar amb claus encriptades.

El següent és un exemple molt senzill en el qual es veu el funcionament de la funció:

```
1 <?php
2 $str = 'apple';
3
4 if (md5($str) === '1f3870be274f6c49b3e31a0c6728957f')
5 {
6     echo "Tens un ordinador Apple";
7 }
8 ?>
```

En el següent exemple es veu com crear una consulta on es comprova si el valor de nom i contrasenya obtingut d'un formulari està a la base de dades:

```
1 $consulta = sprintf('select * from usuaris where nom="%s" and passwd = "%s"',
    $_POST["nom"], $_POST["passwd"]);
```

En el cas que les claus estiguessin encriptades a la base de dades, es podria crear la consulta així:

```
1 $consulta = sprintf('select * from usuaris where nom="%s" and passwd = "%s"',
    $_POST["nom"], md5($_POST["passwd"]));
```

La funció `sprintf()` de PHP funciona de forma semblant a la versió de C. És com un `printf`, però en lloc mostra el resultat per pantalla, el retorna a un string. Us pot resultar molt útil per crear consultes. Podeu veure aquí l'especificació completa de la funció:

www.php.net/manual/en/function.sprintf.php

En aquest cas, recordeu que quan creeu un usuari, la seva clau també haurà d'estar encriptada. Per exemple

```
1 $consulta = sprintf('INSERT into usuaris (nom, passwd, administrador) VALUES
    ("%s", "%s", "%s")', $_REQUEST["nom"], md5($_REQUEST["passwd"]), $_REQUEST["
    administrador"]);
```

3.8.2 Accés restringit a pàgines

Una característica molt habitual dels webs actualment és la de presentar diferents continguts del web depenent de quin sigui el perfil de l'usuari. També hi ha certes pàgines que poden estar protegides per tal d'impedir l'accés als usuaris que no tenen els privilegis necessaris.

Per exemple, un usuari que no està registrat a www.amazon.com pot navegar i veure els productes que estan a la venda. Però no podrà realitzar compres a no ser que estigui registrat, i que hagi iniciat una sessió. D'igual manera, un usuari registrat no podrà accedir a les pàgines d'administració del web si no té els permisos necessaris.

Per implementar aquests permisos d'accés podeu fer servir sessions o emmagatzemar a la base de dades l'estructura de pàgines del web i quins usuaris hi tenen accés.

3.8.3 Seguretat a la base de dades

Les llistes de control d'accés (ACL en anglès *Access Control List*) són el mecanisme de seguretat que utilitza MySQL per a totes les connexions i consultes que els usuaris poden intentar realitzar. Una llista de control d'accés controla si un usuari té permisos per accedir a un objecte. Les sentències GRANT i REVOKE s'utilitzen per controlar l'accés dels usuaris a MySQL. Els permisos poden donar-se en diversos nivells:

- Nivell global: els permisos globals s'apliquen a totes les bases de dades d'un servidor donat. Es guarden a la taula *mysql.user*.
- Nivell de base de dades: els permisos de base de dades s'apliquen a tots els objectes en una base de dades especificada. Es guarden a les taules *mysql.db* i *mysql.host*.
- Nivell de taula: els permisos de taula s'apliquen a totes les columnes en una taula donada. Es guarden a la taula *mysql.tables_priv*.
- Nivell de columna: els permisos de columna s'apliquen a columnes en una taula donada. Es guarden a la taula *mysql.columns_priv*.
- Nivell de rutina: els permisos de rutina s'apliquen a rutines emmagatzemades. Es guarden a la taula *mysql.procs_priv*.

Per exemple, per donar el permís SELECT sobre tota la base de dades a l'usuari existent *francesc*, escriureu la següent comanda:

```
1 GRANT SELECT ON base_dades.* TO francesc@'localhost';
```

Per donar més d'un permís a la vegada, s'han de separar les opcions amb una coma. Així que per atorgar els permisos SELECT, INSERT i DELETE a l'usuari *francesc*, escriureu la següent comanda:

```
1 GRANT SELECT, INSERT, DELETE ON base_dades.* TO francesc@'localhost';
```

Un cop heu acabat heu de refrescar els permisos de la memòria executant la comanda `flush privileges`.

Aquí podeu trobar tota la sintaxi del funcionament de la sentència GRANT:

<http://dev.mysql.com/doc/refman/5.1/en/grant.html>

A continuació es mostra un exemple complet de gestió de permisos d'accés:

Primer entreu a la consola MySQL com a usuari *root* escrivint la següent comanda i a continuació entreu la contrasenya:

```
1 # mysql -p -u root
2 Enter password:
```

Un cop a la consola de comandes de MySQL, creareu la base de dades *base_de_dades* escrivint la següent sentència:

```
1 mysql> CREATE DATABASE base_de_dades CHARACTER SET utf8 COLLATE utf8_bin;
2 Query OK, 1 row affected (0.01 sec)
```

Si ara demanem al MySQL que ens mostri les bases de dades, *base_de_dades* apareixerà a la llista.

```
1 mysql> show databases;
2 +-----+
3 | Database |
4 +-----+
5 | information_schema |
6 | base_de_dades |
7 | mysql |
8 | phpmyadmin |
9 +-----+
10 4 rows in set (0.00 sec)
```

Ara creareu un usuari anomenat *usuari1* amb la contrasenya *123456*.

```
1 mysql> CREATE USER 'usuari1'@'%' IDENTIFIED BY '123456';
2 Query OK, 0 rows affected (0.00 sec)
```

Primer donareu el permís de SELECT a la base de dades *base_de_dades* per l'usuari *usuari1*.

```
1 GRANT SELECT ON base_de_dades.* TO 'usuari1'@'%';
2 Query OK, 0 rows affected (0.00 sec)
```

Per comprovar els permisos d'un usuari es fa servir la comanda `show grants for usuari1`.


```

1 mysql> show grants for usuari1
2   -> ;
3 +-----+
4 | Grants for usuari1@%
5 |
6 +-----+
7 | GRANT USAGE ON *.* TO 'usuari1'@'%' IDENTIFIED BY PASSWORD '*6
  | BB4837EB74329105EE4568DDA7DC67ED2CA2AD9' |
8 | GRANT SELECT ON 'base_de_dades'.* TO 'usuari1'@'%'
  |
9 +-----+
10 2 rows in set (0.00 sec)

```

Heu pogut comprovar com apareix el permís de SELECT, ara afegirem els permisos de DELETE i INSERT a l'usuari *usuari1*.

```

1 mysql> GRANT DELETE,INSERT ON base_de_dades.* TO 'usuari1'@'%;
2 Query OK, 0 rows affected (0.00 sec)

```

Comproveu els privilegis amb la comanda `show grants` i veureu com l'usuari *usuari1* té els privilegis SELECT, INSERT, DELETE a la base de dades *base_de_dades*.

```

1 mysql> show grants for usuari1;
2 +-----+
3 | Grants for usuari1@%
4 |
5 +-----+
6 | GRANT USAGE ON *.* TO 'usuari1'@'%' IDENTIFIED BY PASSWORD '*6
  | BB4837EB74329105EE4568DDA7DC67ED2CA2AD9' |
7 | GRANT SELECT, INSERT, DELETE ON 'base_de_dades'.* TO 'usuari1'@'%'
  |
8 +-----+
9 2 rows in set (0.00 sec)

```

Per acabar i refrescar els permisos de la memòria és necessari executar la comanda `flush privileges`:

```

1 mysql> flush privileges;
2 Query OK, 0 rows affected (0.00 sec)

```

3.9 Verificació i proves

Per verificar el funcionament de MySQL podeu fer servir programari de monitorització que controla la disponibilitat de serveis. Per monitoritzar el servei de base de dades MySQL heu d'instal·lar connectors al programari. Entre les eines de programari de monitorització lliures que existeixen parlarem de:

- Nagios
- Pandora FMS

3.9.1 Nagios

Nagios (www.nagios.com) és un programari molt popular de monitorització de xarxes, ordinadors i aplicacions. Està llicenciat sota la GNU (*General Public License* Version 2 publicada per la Free Software Foundation) i és de codi obert.

Es pot configurar per tal que monitoritzi els equips (maquinari) i serveis (programari) que l'usuari vulgui, alertant quan el comportament d'un recurs no sigui correcte o superi els marges definits per l'administrador de xarxa. Les alertes es poden enviar entre d'altres per correu electrònic i missatges SMS.

Entre les seves característiques principals figuren la monitorització de serveis de xarxa (SMTP, POP3, HTTP, NNTP, ICMP, SNMP, FTP, SSH); de sistemes de maquinari (càrrega del processador, ús dels discs, memòria, estat dels ports, arxius de log, temperatura...); remota mitjançant túnels SSL xifrats o SSH.

També ofereix la possibilitat de programar complements específics per qualsevol paràmetre d'interès d'un sistema.

Podeu trobar diferents *plugins* per monitoritzar bases de dades MySQL i servidors. Podeu trobar-ne un llistat aquí:

<http://exchange.nagios.org/directory/Plugins/Databases/MySQL>

Aquí teniu un llistat d'alguns dels més populars:

- Element de llista de `picscheck_mysql` (<http://oss.isg.inf.ethz.ch/nagiosplug>): comprova i mostra l'estat de les connexions a un servidor MySQL .
- `check_mysql` (<http://william.leibzon.org/nagios>): permet establir llistats per a les variables de la comanda `SHOW STATUS` de MySQL.
- `check_mysql_health` ([http://labs.consol.de /nagios/check_mysql_health](http://labs.consol.de/nagios/check_mysql_health)): permet controlar molts paràmetres d'una base de dades, com ara el temps de connexió, número de connexions, índexs, la taxa d'èxit de la memòria cau, les consultes lentes, etc.

3.9.2 Pandora FMS

Pandora FMS (en anglès *Pandora Flexible Monitoring System*, Sistema de Monitorització Flexible Pandora, <http://pandorafms.com>) és un programari de codi

obert publicat sota llicència GPL2 GNU (*General Public License*), que serveix per monitoritzar i mesurar tot tipus d'elements. Monitoritza d'una forma visual al llarg del temps l'estat de sistemes, aplicacions o dispositius.

Pandora FMS pot recollir informació de qualsevol sistema operatiu, amb programari específic per a cada plataforma: GNU/Linux, AIX, Solaris, HP-UX, BSD/IPS0 i Windows 2000, XP i 2003.

Pandora FMS també pot monitoritzar qualsevol tipus de servei de xarxa com ara: TCP/IP, balancejadors de càrrega, *routers*, *switches*, sistemes operatius, aplicacions o impressores.

Per verificar el funcionament de MySQL existeixen mòduls del propi programari i connectors desenvolupats. Entre els mòduls trobem:

- MySQL Monitoring: realitza un seguiment general de MySQL.
- Mysql Cluster manager process: permet fer un seguiment dels processos de MySQL.
- Mysql Active Connections: permet verificar les connexions actives de MySQL.

3.9.3 Proves de rendiment

Una cosa molt important en qualsevol desenvolupament web són les proves de rendiment. Amb elles podreu avaluar atributs quantitatius i qualitius del vostre sistema com la fiabilitat o la escalabilitat, ja siguin a nivell de programari o maquinari. Dins de les proves, les bases de dades han de ser una cosa fonamental.

Mysqslap és una eina de diagnòstic de MySQL disponible des de la versió 5.1.4 que us permetrà realitzar aquestes proves. Mysqslap emula la càrrega d'un gran nombre de clients accedint a un servidor MySQL i ofereix informes una vegada acabada la prova.

Mysqslap pot generar sentències SQL automàticament si no es poden utilitzar les sentències que executa el vostre sistema.

Per invocar mysqslap ho podeu fer d'aquesta manera:

```
1 shell> mysqslap [opcions]
```

Podeu trobar aquí un llistat de les diferents opcions de mysqslap:

<http://dev.mysql.com/doc/refman/5.1/en/mysqslap.html>

Mysqslap s'executa en tres etapes: preparació, execució i neteja. A la preparació es creen les taules i les consultes per la prova. A l'execució es fan servir més d'una connexió de clients per executar la prova. A la neteja s'eliminen les taules si s'especifica i es desconnecten els clients.

Un exemple el tenim a:

Especifiqueu la consulta SQL (`SELECT * FROM taula`) i creeu la taula (`CREATE TABLE taula (clau int);INSERT INTO taula VALUES (365)`), especifiqueu 50 clients i 200 consultes seleccionades per cadascun:

```
1 shell> mysqlslap -uroot -p --delimiter=";" \
2   --create="CREATE TABLE taula (clau int);INSERT INTO taula VALUES (365)" \
3   --query="SELECT * FROM taula" --concurrency=50 --iterations=200
```

Un cop entrat la contrasenya de l'usuari *root* mostrarà unes estadístiques amb els temps (mitjà, mínim i màxim) d'execució de totes les consultes:

```
1 Benchmark
2   Average number of seconds to run all queries: 0.021 seconds
3   Minimum number of seconds to run all queries: 0.009 seconds
4   Maximum number of seconds to run all queries: 0.900 seconds
5   Number of clients running queries: 50
6   Average number of queries per client: 1
```

Especifiqueu que mysqlslap creï la instrucció de consulta SQL a una taula de dues columnes INT i tres columnes VARCHAR. Utilitzeu cinc clients de consulta 20 vegades cada un. No creeu la taula amb la consulta per inserir les dades (és a dir, s'usarà l'esquema de la prova anterior i les dades):

```
1 shell> mysqlslap -uroot -p --concurrency=5 --iterations=20 \
2   --number-int-cols=2 --number-char-cols=3 \
3   --auto-generate-sql
```

Un cop entrada la contrasenya de l'usuari *root*:

```
1 Enter password:
2 Benchmark
3   Average number of seconds to run all queries: 0.016 seconds
4   Minimum number of seconds to run all queries: 0.009 seconds
5   Maximum number of seconds to run all queries: 0.106 seconds
6   Number of clients running queries: 5
7   Average number of queries per client: 0
```

Creareu dos arxius un de creació (`creacio.sql`) amb les sentències (`CREATE TABLE` i `INSERT INTO`) i un altre de consultes (`consultes.sql`) amb sentències (`SELECT`). Primer executarà l'arxiu de creació i després s'executarà l'arxiu amb les consultes. Especificareu cinc clients, cinc vegades cadascun:

```
1 mysqlslap -uroot -p --concurrency=5 \
2   --iterations=5 --query=query.sql --create=create.sql \
3   --delimiter=";"
```

Resultat:

```
1 Enter password:
2 Benchmark
3   Average number of seconds to run all queries: 0.037 seconds
4   Minimum number of seconds to run all queries: 0.037 seconds
5   Maximum number of seconds to run all queries: 0.037 seconds
6   Number of clients running queries: 5
7   Average number of queries per client: 0
```