Programmare Applicativi Web in PHP e Javascript

Francesco Fiorà

Versione 1.1_1
10 Ottobre 2009

Copyright @ 2009 Francesco Fiorà – www.francescofiora.it
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 2 or any later version published by the Free Software Foundation.
Read the entire license text here: http://www.gnu.org/licenses/gpl.html.

I marchi sono proprietà dei loro possessori.
Nonostante le informazioni date in questo documento si pensa siano corrette, l'autore non accetterà responsabilità per il contenuto di questo documento. Usate i suggerimenti e gli esempi qui contenuti a vostro rischio.

Sommario

Introduzione

Sviluppare applicativi web, non è più difficile di quello che sembra, l'unico piccolo aspetto negativo che caratterizza il processo di sviluppo è dato dall'uso di diverse tecnologie distinte, ma nello stesso tempo in relazione fra loro. Tralasciando l'aspetto progettuale (cioè l'elaborazione di specifiche e diagrammi), un applicativo nasce con la bozza del layout elaborato su un programma di grafica. Per la produzione fisica del layout viene utilizzato l'HTML, e quindi anche i fogli di stile. Per la creazione delle pagine dinamiche vengono utilizzati due linguaggi di programmazione, uno lato server (come il PHP) e l'altro lato client (javascript). Per dialogare con il gestore di database (come MySql), viene utilizzato il linguaggio SQL. Nelle pagine dinamiche possono anche esserci delle applet Java, dei controlli in Flash, o anche dei controlli in Ajax.
Fortunatamente esiste la diversificazione delle carriere professionali che consente ai programmatori web di concentrarsi solo sui linguaggi di programmazione, e ai grafici web, di concentrarsi sui programmi di grafica e su Flash. Ovviamente entrambi devono conoscere perfettamente l'HTML e i fogli di stile.
Le applet Java solitamente sono poco utilizzate, e nella maggior parte dei casi, in Internet si trova tutto quello di cui si ha bisogno.
Questo documento copre diversi argomenti riguardanti la programmazione in PHP, Javascript ed SQL ed è stato scritto grazie alla mia esperienza di programmatore, e responsabile software. In particolare gli argomenti trattati, riflettono la mia esperienza di affiancamento dei nuovi programmatori.
I vari contenuti, sono molto utili a tutti quei programmatori che hanno una minima esperienza nella programmazione web. In particolare viene offerto al lettore, la possibilità di riprendere porzioni di codice da usare e modellare a piacere.
Ci sono alcuni prerequisiti per poter utilizzare questo documento. In primo luogo occorre avere una discreta conoscenza dell'HTML e dei fogli di stile**.** È impensabile sperare di poter programmare in PHP e sviluppare gestionali complessi, utilizzando editor visuali. In secondo luogo occorre conoscere le basi dei linguaggi PHP, javascript e SQL. Per potervi esercitare dovrete installare sul vostro computer PHP e MySql (possibilmente le versioni più recenti).
La maggior parte del contenuto del documento è composto da codice sorgente, in particolare viene utilizzata una tecnica molto semplice per spiegare sorgenti abbastanza complessi: si parte con un semplice esempio, ed ogni argomento successivo viene aggiunto del codice al sorgente, fino ad arrivare, passo dopo passo ad un codice ricco di funzionalità. Ogni passo è preceduto da una spiegazione concisa e spesso seguita da osservazioni.

Cosa c'è di nuovo

Rispetto alla versione precedente, sono state effettuate alcune correzioni, in oltre ci sono diverse aggiunte: installare applicativi PHP su penna USB o CD-ROM; implementazione a oggetti dei menu statici; stampa in HTML della rubrica; inviare e-mail; un metodo per archiviare dei dati statici.

Panoramica

Nel primo capitolo si parte con delle spiegazioni rapide di alcuni trucchi e istruzioni, utili e spesso necessari per il corretto sviluppo di un progetto. Poi viene introdotto un metodo rapido per installare un server Apache+MySQL+PHP su Hard-Disk, CD-ROM e Penna USB per Windows. Successivamente viene illustrata una breve spiegazione su come organizzare i file di un progetto, ed infine una breve illustrazione del contenuto dei sorgenti allegati al documento.

Il secondo capitolo mostra diversi aspetti fondamentali del linguaggio SQL, tra cui: le differenze tra chiavi e indici, l'operatore JOIN, le sub-query, le transazione protette ed altro ancora.

Il terzo capitolo si occupa di illustrare le tecniche standard di programmazione. Si parte con i primi brevi esempi su come implementare semplici menu statici. Si continua con la spiegazione di come implementare la stampa di una tabella con filtri e ordinamenti. Si conclude con la spiegazione di come implementare un FORM di inserimento/modifica con validazione lato client.

Il quarto capitolo mostra vari moduli accessori che un applicativo potrebbe avere, come caricare un documento tramite upload del file, creare pagine stampabili in HTML, esportare o importare in diversi formati quali CSV, Excel e XML, ed inviare una e-mail.

Il quinto capitolo da una spiegazione completa sull'implementazione dell'autenticazione, partendo da quella più semplice, fino a quella più complessa con restrizioni di accesso e cronologia degli accessi.

Il sesto capitolo si occupa di tecniche avanzate di programmazione come un metodo per archiviare dati statici, l'implementazione dei FORM dinamici, ed alcuni effetti dinamici mediante l'utilizzo di javascipt, come il trascinamento di un contenitore.

Il settimo capitolo mostra come popolare un database di un gestionale con dati di prova, per agevolare la cruciale fase di alfa-testing durante l'implementazione.

Introduzione alla programmazione

Questo capitolo dettaglia le problematiche del linguaggio PHP che meritano una particolare attenzione, in quanto si è partiti dal presupposto che il lettore conoscesse solo le basi di questo linguaggio di programmazione. Verranno puntualizzati alcuni concetti sulle stringhe, sulla connessione a database e sulla gestione delle variabili provenienti da GPC (Get/Post/Cookie). In oltre è stato dato anche un accenno su come organizzare i file di un progetto, in modo da orientare il programmatore con poca esperienza nella direzione corretta nella progettazione di un programma.

Pillole

Differenze tra virgolette e apici

Quando si devono usare le stringhe, si possono commettere molti errori se non si conoscono bene le differenze tra l'uso delle virgolette e quelli degli apici. Vediamo due esempi e i rispettivi risultati, per quanto riguarda l'inclusione di variabili.

Le istruzioni

```
$testo='ciao';
echo 'testo = "$testo" ';
```

daranno come output:

```
testo = "$testo"
```

Mente le istruzioni

```
$testo='ciao';
echo "testo = \"$testo\" ";
```

daranno come output:

```
testo = "ciao"
```

Eco altri due esempi sulla stampa dei caratteri speciali.

```
echo 'ciao\nciao';
```

dà come output:

```
ciao\nciao
```

Mente l'istruzione

```
echo "ciao\nciao";
```

dà come output:

```
ciao
ciao
```

Cattive abitudini con i vettori

Il motivo per cui è sconsigliato scrivere $vett[pippo] al posto $vett['pippo'], è che nel primo caso il PHP inizialmente considera pippo una costante, quindi se essa non risulta definita gli assegna come stringa il suo nome cioè 'pippo'. Oltre alle operazioni in più che il PHP è costretto a eseguire, esiste anche il rischio di utilizzare costanti che sono realmente definiti dal programma.

Connessione a MySQL

Un'applicazione web ha solitamente un file "connessione.php" che contiene le seguenti istruzioni per la connessione a MySql

```
<?php
$db = mysql_connect("server", "username", "password") or die ("Errore nella connessione");
mysql_select_db("nome_database", $db) or die ("Errore nella selezione del database.");
?>
```

Ipotizzando che il file "connessione.php" risieda nella cartella include, ad ogni file php che avrà necessità di collegarsi a database gli basterà eseguire le seguenti istruzioni

```
include('include/connessone.php');
```

Uso di @ come operatore di controllo errore

Se ad una espressione viene inserito come prefisso il carattere @, allora qualunque messaggio di errore di quella espressione, sarà ignorato. È possibile recuperare l'ultimo messaggio di errore generato tramite la variabile $php_errormsg (se nel PHP.INI la proprietà track_errors è abilitata).

Gestione degli errori

Nel corso del libro per eseguire una query verrà utilizzato spesso questa istruzione

```
mysql_query($query) or errore_db($query);
```

In caso di errore, viene richiamata la funzione errore_db().
L'output degli errori di un programma dipende dalla fase di sviluppo di quest'ultimo. Se il programma si trova in versione alfa, vorremmo che l'output venga visualizzato immediatamente, mentre nella versione beta, l'output deve essere archiviato in un apposito file.

```
function errore_db($query)
{
    global $parametri;
    if ($parametri['debug'] != 'no') echo "<br>Errore nella query: $query <br>".mysql_error();
    if ($parametri['file_errori'])
    {
        $f = @fopen($parametri['file_errori'], 'a');
        @fwrite($f, "\n\n".date('d/m/Y')."\nErrore nella query: $query \n".mysql_error());
        @fclose($f);
    }
    exit();
}
```

Si osservi che se il vettore $parametri viene definito, la funzione restituisce gli errori in output e li archivia su file.
Ovviamente, conviene inserire quanta funzione all'interno del file "connessione.php".

Caricare una riga di un risultato su un array

La funzione mysql_fetch_array() (di defautl) carica una riga del risultato di una query come un array associativo e numerico. Con l'opzione MYSQL_ASSOC, restituisce un array associativo esattamente come la funzione mysql_fetch_assoc(), mentre con l'opzione MYSQL_NUM restituisce un array numerico esattamente come la funzione mysql_fetch_row(). L'opzione di default è MYSQL_BOTH che restituisce entrambi gli array. Una funzione particolare è mysql_fetch_object() che restituisce un oggetto al posto di un array. Si preferisce spesso utilizzare la funzione mysql_fetch_array() senza opzioni, perché di fatto rende il sorgente più versatile.

register_globals

Nel file PHP.INI se il parametro register_globals indica se registrare o meno le variabili EGPCS (Environment, GET, POST, Cookie, Server) come variabili globali.
Ad esempio, se la variabile $nome proviene da un FORM (con METHOD a POST), e register_globals e settato a on allora occorre eseguire la seguente istruzione.

```
$nome=$_POST['nome'];
```

Anche se dalla distribuzione PHP 4.2.0 in poi, register_globals viene settato per defautl a off, molte aziende che offrono spazio web, hanno impostato questo parametro a on per consentire la compatibilità con i vecchi siti. Si noti però, che dalla distribuzione del PHP 6, questo parametro sarà rimosso.

magic_quotes_gpc

Nel file PHP.INI se il parametro magic_quotes_gpc è impostato a on, allora le variabili provenienti da GPC (Get/Post/Cookie) avranno tutti i ' (apici singoli), " (doppi apici), \ (backslash) e NULL preceduti in automatico dal backslash.
Questo parametro è stato aggiunto per ridurre il rischio di scrivere codice pericoloso da parte dei principianti. Se si disabilita magic-quote, occorre prestare molta attenzione a proteggersi dagli attacchi di SQL injection. Questo concetto risulterà chiaro con un esempio. Ipotizziamo di voler visualizzare la lista dei documenti dell'utente connesso (identificato tramite la variabile $id_utente), e di voler filtrare la ricerca mediante il nome del documento. Aspettandoci di riceve la variabile $nome dal FORM di ricerca, otteniamo il seguente codice

```
$nome=$_POST['nome'];
$query = " SELECT * FROM documenti WHERE nome LIKE '%$nome%' AND id_utente= '$id_utente' ";
mysql_query($query) or errore_db($query);
```

Se magic_quotes_gpc è impostato a off, e la variabile $nome contiene un apice, la query torna un errore, ed ancora peggio, se $nome contiene ' OR (1=1) OR "<>' allora l'utente è in grado di visualizzare tutti i documenti di tutti gli utenti.
Il codice andrebbe riscritto in questo modo

```
$nome=$_POST['nome'];
$nome = addslashes($nome) ;
$query = " SELECT * FROM documenti WHERE nome LIKE '%$nome%' AND id_utente= '$id_utente' ";
mysql_query($query) or errore_db($query);
```

Quando la stessa variabile deve essere stampata in output e magic_quotes_gpc è impostato a on allora occorre eseguire uno stripslash(). Di seguito l'esempio

```
<input name="nome" value="<?php echo stripslash($_POST['nome']); ?>">
```

Di default magic_quotes_gpc viene settato a on, ma per scrivere applicativi compatibili, occorre utilizzare get_magic_quotes_gpc() che restituisce il valore impostato in PHP.INI; tramite quest'ultima è possibile decidere quando utilizzare addslashes() o stripslashes().

Ecco due semplici funzioni che aiutano il programmatore.

```
function post_a_sql($valore)
{
   if (get_magic_quotes_gpc()==0)
   {
      return is_array($valore) ? array_map('post_a_sql', $valore) : addslashes($valore);
   }
   return $valore;
}

function post_a_uso($valore)
{
   if (get_magic_quotes_gpc()==1)
   {
      return is_array($valore) ? array_map('post_a_uso', $valore) : stripslashes($valore);
   }
   return $valore;
}
```

La prima funzione viene utilizzata per le variabili destinate alla creazione di query, mentre la seconda per il normale output o anche per utilizzi operativi. Si noti che funzionano anche con le matrici.
Conviene avere queste funzioni all'interno di un file "funzioni.php".
L'esempio seguente farà capire meglio quanto detto prima. Si provi a scrivere testi contenenti apici e a variare nel file php.ini il parametro magic_quotes_gpc

```
<html>
<body>
<?php
include('funzioni.php');
echo "<br>magic_quotes_gpc = ".get_magic_quotes_gpc();
$test = $_POST['test'];
echo "<br>test = $test";
echo "<br>addslashes(test) = ".addslashes($test);
echo "<br>stripslashes(test) = ".stripslashes($test);
echo "<br>post_to_sql(test) = ".post_a_sql($test);
```

```
echo "<br>post_to_use(test) = ".post_a_uso($test);
?>
<br>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input name="test" value="<?php echo $test; ?>"> <input type="submit">
</form>

</body>
</html>
```

Un metodo pratico e veloce è quello di avere a disposizione due vettori per ogni esigenza

```
$_POST_SQL = array_map('post_a_sql', $_POST);
$_GET_SQL = array_map('post_a_sql', $_GET);
$_COOKIE_SQL = array_map('post_a_sql', $_COOKIE);

$_POST = array_map('post_a_uso', $_POST);
$_GET = array_map('post_a_uso', $_GET);
$_COOKIE = array_map('post_a_uso', $_COOKIE);
```

Ovviamente conviene utilizzare questa tecnica solo quando effettivamente si ha bisogno di convertire tutte le variabili.

htmlspecialchars

Questa funzione si utilizza per prevenire la presenza di marcatori HTML negli input utente. Vediamo questo esempio:

```
<html>
<body>
<?php
include('include/funzioni.php');
$test = post_a_uso($_POST['test']);
echo "<br>test = $test";
echo "<br> htmlspecialchars (test) = ".htmlspecialchars($test);
?>
<br>
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input name="test" value="<?php echo $test; ?>"> <input type="submit">
</form>

</body>
</html>
```

Proviamo ad inserire nel campo input la parola "ciaociao" e vediamo l'output:

```
test = ciao ciao
htmlspecialchars (test) = <b>ciao</b> ciao
```

Per evitare visualizzazioni errate (come ad esempio chiusure dei tag accidentali) o inserimenti malevoli di codice javascript, è consigliato l'uso di questa funzione.

urlencode e htmlentities

Quando si crea un link e occorre passare dei parametri può essere conveniente utilizzare la funzione urlencode per codificare i valori dei parametri. Ad esempio, se in un link occorre passare la variabile $var1 contenente il valore "ciao & ciao" e non viene utilizzata questa funzione, la variabile non viene passata in modo corretto; invece con urlencode il valore viene convertito in "ciao+%26+ciao".
L'uso della funzione htmlentities è legata alle specifiche dello standard W3C, dove ad esempio il carattere "&" deve essere sostituito con "&". Vediamo un esempio sull'uso di queste funzioni

```
<?php
```

```
$link = 'var1='.urlencode('1').'&var2='.urlencode('ciao & ciao');
?>
<a href="index.php?<?=htmlentities($link)?>">test</a>
```

Da cui si ottiene il seguente codice HTML

```
<a href="index.php?var1=1&amp;var2=ciao+%26+ciao">test</a>
```

Da notare che se occorre inviare un solo parametro contenente un numero, allora l'uso di queste funzioni può anche essere omesso.

Installare applicativi PHP su penna USB o CD-ROM

Chi ha un minimo di dimestichezza con questo linguaggio dovrebbe essere già in grado di installare e configurare PHP/Apache/MySql sul proprio computer; tuttavia ho ritenuto opportuno illustrare la possibilità di installare tali servizi su penna USB o su CD-ROM.
Prima di procedere, ci sono alcuni concetti chiave che devono essere illustrati:

- In tutti e due i supporti, devono essere presenti PHP/Apache/MySql.
- Ognuno di questi applicativi deve lavorare con file di configurazione creati ad hoc al momento dell'esecuzione in quanto tutti i percorsi cambiano da computer a computer (soprattutto il drive dove si trova il supporto).
- Le porte sui quali lavorano Apache e MySql devono essere diversi da quelli soliti, per evitare conflitti nel caso in cui tali supporti fossero inseriti su computer con già installo Apache e MySql.
- Nel supporto CD-ROM, i database di MySql devono essere copiati sull'hard-disk locale prima di poterli utilizzate. Alla chiusura di MySql tali database devono essere eliminati.

Tutte le problematiche elencate sopra possono essere agevolmente superate utilizzando applicativi ad hoc, come ad esempio Server2Go. Quest'ultimo è un applicativo Open Source e può essere scaricato dal sito ufficiale: www.server2go-web.de

Hard Disk

Una volta scaricato l'archivio potete scompattatelo nel supporto dove volete installarlo. Quindi dovete solo configurare il file pms_config.ini
Ecco un esempio di configurazione (i commenti sono stati tradotti)

```
[general]
;--- Se usi il parametro ShowTrayIcon, puoi definire un nome che sarà utilizzato dal Menu e dal Tooltip
ServerName=Server2Go
;--- Mostra uno splash screen allo startup. Se qui non c'è un valore, non verrà mostrato lo splash. Puoi impostare
;--- un file grafico. I formati grafici al momento sono:  BMP GIF JPG PNG TIF ICO TGA PCX PSD
UseSplash=splash.png
;--- Mostra la barra di progressione allo startup
ShowStartupProgress=1
;--- Puoi impostare un colore che sarà mostrato trasparente. Puoi definire un valore RGB
;--- nello stile SplashTransparencyColor=128,128,128
SplashTransparencyColor=128,128,128
;--- Imposta questo a vero se vuoi terminare l'esecuzione del server dopo aver chiuso il browser o se non usi il tipo
;--- di browser predefinito. Nota che se impostato a vero l'unico modo per effettuare lo shutdown del server
;--- è uccidere il suo processo attraverso il taskmanager
KeepRunningAfterBrowserClose=1
;--- Se il succesivo parametro è 1 Server2Go mostra una tray icon per permettere lo shutdown del server.
;--- Se non ti piace la tray icon imposta il parametro con valore 0
ShowTrayIcon=1
;--- Server2Go non accede in scrittura nei file di configurazione dei server. Per questo tutti i file di configurazione
;--- saranno copiati in una cartella temporanea
;--- Imposta StartLocal a 0 se vuoi usare Server2Go in un ambiente scrivibile  (es. Harddisk, Penna USB, …)
;--- altrimenti puoi impostare il valore 1
;--- Nota che StartLocal=1 NON LAVORA su un CD-ROM!!!!!
StartLocal=1
;--- Puoi restringere il numero di esecuzioni di istanze dell'applicativo ad una istanza. Questo vuol dire puoi
;--- utilizzare solo una istanza della tua applicazione web basata su server2go. Se imposti il seguente parametro
;--- a 1 solo una istanza potrà essere eseguita
AllowOnlySingleInstance=1
;--- Metti il messaggio di shutdown che sarà mostrato quando Server2Go si sta chiudendo, {SERVER_NAME} è
```

```
;--- sostituito dal nome  impostato all'inizio del file
ShowShutdownMessage=Shutdown {SERVER_NAME}


[http]
;--- Definisci l'hostname che il browser mostrerà come url e che verrà utilizzato per apache server
;--- Al momento è possible inserire solo l'indirizzo IP locale.
HostName=127.0.0.1
;--- Definisci la porta che verrà utilizzata. Se la porta è già in uso verrà cercata una porta libera automaticamente
Port=80
;--- La pagina html/php di partenza, così puoi definire una pagina in tua cartella come file di partenza
;--- (es. pages/start.php)
;--- Se vuoto verrà usato quella di default (index.php, index.html, index.htm...)
DefaultFile=
;--- definisci la cartella in cui tutti i file web sono salvati
DefaultRoot=htdocs
;--- Se il valore è 1 tutti i file temporanei del server http (sessions...) saranno cancellati dopo lo shutdown
DeleteHttpTemp=1
;--- Se HideTempFolder è impostato a 1 la cartella temporanea verrà creata come cartella nascosta
HideTempFolder=1


[Browser]
;--- Il browser che parte dopo la partenza del server (possibili valori: IEXPLORER, FIREFOX,
PORTABLEFIREFOX, MOZILLA, DEFAULT, EXTERNAL)
;--- Guardare la documentazione alla pagina www.server2go-web.de/wiki
;--- su come utilizzare PORTABLEFIREFOX!!!!!!!!
BrowserType=IEXPLORER
;--- comandi opzionali del Browser (NON IN USO AL MOMENTO)
BrowserCommandOptions=
;--- Percorso del browser esterno
BrowserPath=ExternalBrowser/SimpleBrowser.exe
WorkOfflineTitle=
;--- La dimensione della finestra del browser. Può essere una dimensione in pixel (1024x768)
;--- o valori come MAXIMIZE e KIOSK (solo Internet Explorer)
BrowserSize=


[database]
;--- 1 se MySQL Server deve partire
UseMySQL=1
;--- 1 se i file dei database della cartella dbdir sarà spostata in una cartella della macchina locale
LocalMirror=0
;--- 1 se lo spostamento del database sarà sovrascritta ogni volta che il server parte
OverwriteLocalMirror=0
;--- Il percorso dove spostare il database (es. c:\MyS2GApp\Data\) , se vuoto verrà utilizzata la cartella di default
MirrorFolder=
;--- se il valore è 1 tutti i file del database saranno eliminati dopo lo shutdown del server
DeleteDatabaseFiles=0
;--- La porta che MySQL utilizzerà. Se vuota mysql utilizzerà la porta di default
MySQLPort=
;--- Parametri a linea di comando (es. skip-innodb)
MySQLCmd=
```

```
;--- se HideMirrorFolder è impostato a 1 la cartella verrà creata come cartella nascosta
HideMirrorFolder=0

[AdditionalParsing]
;--- Tu puoi definire dei file addizionali che verranno parsati allo startup per sostituire le stringhe.
;--- Puoi utilizzare esso cioè aggiungere il percorso del server corrente al file htaccess
;--- o cambiare la configurazione dei file della tua applicazione web allo startup
;--- NOTE THAT THIS WILL ONLY WORK IF YOU ARE NOT WORKING ON A WRITE PROTECTED MEDIUM LIKE A CD!!!!!
;--- Utilizza questi in un harddisk or penna usb...
File1=
File2=
File3=
File4=
File5=
File6=
File7=
File8=
File9=
File10=

[Path]
```

L'esempio dato sopra riguarda la configurazione per una installazione su computer per uso sviluppo. I server utilizzano le porte di default, l'avvio avviene mandando in esecuzione server2go.exe e lo shotdown può avvenire mediante utilizzo della icon tray. Con questa configurazione i server occupano risorse solo se si ha intenzione di utilizzarli.

Penna USB

L'installazione su penna USB è simile alla precedente, di seguito vediamo le sole opzioni che differiscono.

```
[general]
KeepRunningAfterBrowserClose=0
ShowTrayIcon=0

[http]
Port=4001

[database]
MySQLPort=7188
```

L'esempio dato sopra riguarda la configurazione per una installazione su dischi rimovibili quindi destinato ad un uso 'occasionale' su computer diversi. I server utilizzano porte diverse da quelle di default per evitare possibili conflitti, l'avvio avviene mandando in esecuzione server2go.exe e lo shotdown avvenire chiudendo il browser.

CD-ROM

L'installazione in un supporto a sola lettura è un po' più complessa rispetto a quanto visto in precedenza.
Il primo passo consiste nel predisporre una installazione simile a quella della penna USB.
Il secondo passo consiste nel creare e riempire i vari i database che necessita il vostro applicativo; ricordatevi che potete farlo solo adesso.
Il terzo passo consiste nel modificare il file pms_config.ini in modo che il server lavora in un ambiente a sola lettura, cioè tutti i file di database e tutti i file temporanei verranno spostati o creati nella cartella locale.

```
[general]
KeepRunningAfterBrowserClose=0
ShowTrayIcon=0
StartLocal=1

[http]
Port=4001

[database]
MySQLPort=7188
LocalMirror=0
OverwriteLocalMirror=0
DeleteDatabaseFiles=1
```

Adesso, potete creare un file ISO del vostro archivio. Per una verifica del il corretto funzionamento potete utilizzare un qualunque programma di gestione di cd virtuali.
Questo esempio riguarda la configurazione per una installazione su CD-ROM destinato ad un uso 'occasionale' su computer diversi.

Si osservi che giocando con il file di configurazione, potete creare un gestionale dove i file del database risiedono su una cartella locale, mentre l'applicativo gira su CD-ROM (ovviamente questa procedura è di scarsa utilità).

Organizzare i file in cartelle

Una buona organizzazione dei file in cartelle costituisce la base per una buona progettazione di un programma.
Un sito molto semplice (con connessione a database) organizza le cartelle in modo simile a quello seguente

- **include**: contiene file php, essi costituiscono una libreria di funzioni e parametri del programma
- **img**: immagini e file css
- **javascript**: file javascript
- **./**: file php principali
- **doc**: eventuali documenti o foto
- **parametri**: contiene i file php di configurazione del programma

La cartella 'include' contiene solitamente i seguenti file:

- **funzioni.php:** contiene le funzioni più utilizzate dal programma;
- **top.php:** contiene il codice HTML della parte superiore della pagina (o eventualmente anche la parte sinistra);
- **bottom.php:** contiene il codice HTML della parte inferiore della pagina;

Questa cartella potrebbe contenere anche altri file utili al programma.

La cartella 'parametri' contiene solitamente i seguenti file:

- **parametri.php**: contiene le variabili utili nel programma come percorsi e impostazioni di base;
- **connessione.php**: contiene le istruzioni relative alla connessione a database.

Questi file sono tenuti in una cartella separata dal resto del programma in modo da facilitare i successivi aggiornamenti, infatti per l'aggiornamento di una installazione è sufficiente sostituire tutti i file e cartelle ad esclusione della cartella parametri.

Quando si implementa un gestionale molto complesso ovviamente si cerca di implementare una struttura modulare, che si rifletterà di conseguenza sulla struttura delle cartelle. Ci sarà una cartella per ogni modulo, e queste ultime risiederanno nella cartella (moduli). Per ogni libreria javascript (composta da più file) ci sarà una cartella corrispondente, e quest'ultima risiederà nella cartella (javascript). Se ci saranno librerie a oggetti in php, essere risiederanno nella cartella (classi).
Quando si inizia a sviluppare un gestionale, conviene gestire i percorsi mediante delle costanti gestiti dal file parametri.php. Di seguito un esempio

```
<?php
define ('percorso_principale', '/user/var/www/miosito/');
define ('url_principale', 'http://127.0.0.1/miosito/');
?>
```

A questo punto in una pagina PHP un'immagine verrebbe scritta in questo modo

```
<img src="<?=url_principale?>img/test.gif">
```

Il vantaggio è che in qualunque cartella mettiamo la pagina PHP, l'immagine verrà visualizzata correttamente. La stessa cosa può essere fatta con le inclusioni delle librerie PHP.

Vediamo di seguito, il codice di un possibile file 'top.php'

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Titolo del sito</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script type="text/javascript" src="<?=url_principale?>javascript/script.js"></script>
<link rel="stylesheet" href="<?=url_principale?>img/stile.css" type="text/css">
</head>
```

```
<body style="margin:0">
```

Un file 'top.php', può anche contenere il menu del programma, e la sua implementazione verrà ripresa nel capitolo successivo.
Il codice seguente riguarda un semplice esempio di un possibile file 'bottom.php'

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

Normalmente un file 'bottom.php' contiene la normale chiusura dei tag, aperti dal file 'top.php'.
Un file nella cartella principale potrebbe essere implementato in questo modo:

```
<?php
include('parametri/parametri.php');
include(percorso_principale.'parametri/connessione.php');
include(percorso_principale.'include/funzioni.php');
include(percorso_principale.'include/top.php');
?>
……
……
<?php
include(percorso_principale.'include/bottom.php');
?>
```

Si noti che per spostare una tale pagina in un'altra cartella, occorre solamente cambiare il percorso dell'inclusione del file parametri.php.
Quando ci sono numerosi file da includere, è utile avere un file implementato al solo scopo di includere tutti i file necessari. Un esempio classico è il file che include tutte le classi relative ai controlli HTML. Un simile file "classi.php" potrebbe essere implementato in questo modo

```
<?php
include(percorso_principale.'classi/html/tag.php');
include(percorso_principale.'classi/html/vista_albero.php');
include(percorso_principale.'classi/html/menu.php');
include(percorso_principale.'classi/html/pagina.php');
?>
```

Quando occorre inserire una nuova classe nel progetto, basta aggiungere una riga di codice nel file "classi.php".

Esempi del documento

In alcuni casi, gli esempi inseriti in questo documento riportano solo alcune porzioni di codice, per visualizzare l'interno sorgente si può fare riferimento ai sorgenti forniti a parte. Se non li avete, potete trovare l'intero archivio all'indirizzo http://www.francescofiora.it
Per installare gli esempi copiare la cartella 'esempi' (potete anche cambiare questo nome) nella vostra cartella 'htdocs', quindi procedere con la configurazione seguente.

- Creare con il PhpMyAdmin due database uno chiamato 'esempio' e un altro chiamato 'esempio2' (questi nomi possono essere cambiati);
- Popolare i database creati con i file 'esempio.sql' ed 'esemio2.sql';
- Configurare i seguenti file:
  - 'parametri/connessione.php': accesso al database esempio;
  - 'parametri/connessione2.php' accesso al database esempio2;
  - 'parametri/parametri.php': percorso fisico e indirizzo HTTP della cartella degli esempi.
- Dare il permesso in scrittura alla cartella 'capitolo4' in quanto ci sono degli esempi di esportazione file.

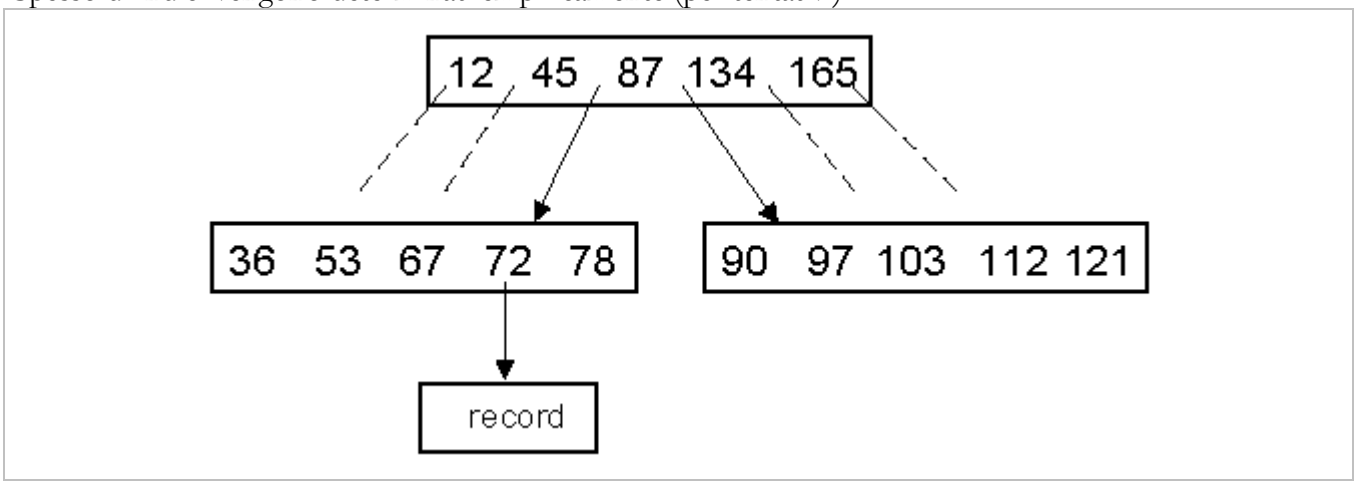
SQL

Questo capitolo si concentra su alcuni aspetti fondamentali del linguaggio SQL, in particolare verranno spiegati diversi argomenti riguardanti le interrogazione a database tramite il PHP.
Si parte dal presupposto che il lettore conoscesse le basi del linguaggio SQL, quindi sia in grado di implementare semplici query di visualizzazione, inserimento, modifica ed eliminazione. Quindi ci si concentra su aspetti più complessi ma soprattutto molto utili nella programmazione.
Gli argomenti che verranno affrontati direttamente legati al linguaggio SQL sono principalmente la differenza tra chiavi e indici, la proiezione di più tabelle con l'operatore JOIN, l'uso degli alias, le sub-query, le transazione protette e l'ottimizzazione delle interrogazioni.

Chiavi e indici

Dallo schema E-R alle tabelle

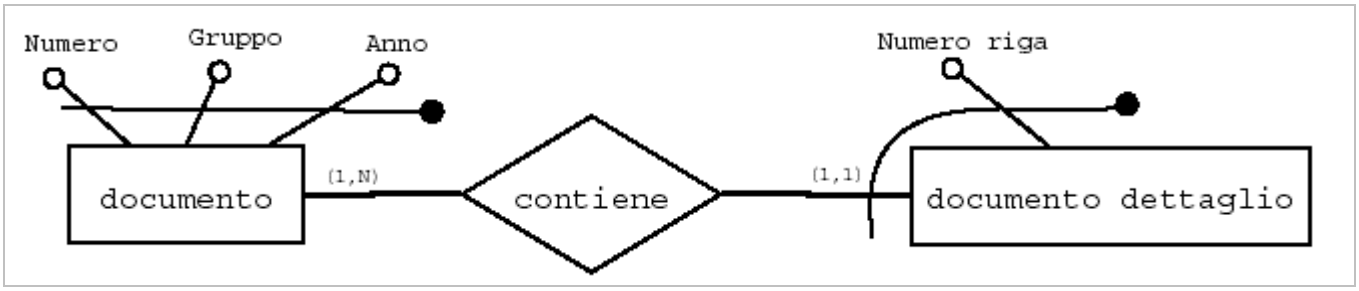
Dopo aver sviluppato una basi di dati mediante lo schema Entità Relazione, ci si trova a dover implementare le tabelle fisiche. Il grosso del lavoro successivo, corrisponde a impostare opportunamente gli indici. Prima di parlare degli indici vediamo alcune considerazioni sulle chiavi create nella progettazione.
La chiave primaria è quel campo che assicura l'univocità del record: assenza di valori nulli e valori univoci.
Poiché ci si aspetta che la maggior parte degli accessi fanno riferimento ad essa, il gestore di database crea un albero di ricerca su di essa. In pratica, quando occorre cercare un record tramite una chiave primaria, invece di cercarlo leggendo i record successivamente, lo cerca in una struttura dati ad albero, partendo dal nodo radice e leggendo successivamente solo i nodi figli che portano alla chiave, riducendo notevolmente i tempi di ricerca. Quando trova il nodo foglia relativo alla chiave, in esso ci sarà un puntatore al record effettivo, dove andarlo a leggere. Un campo impostato come indice permette di creare un altro albero di ricerca. L'uso di indici, quindi serve per effettuare delle query più veloci ma aumentano il volume dei dati archiviati e il tempo impiegato per gestire gli alberi di ricerca nelle operazioni di inserimento e modifica. Spesso di indici vengono determinati empiricamente (per tentativi).



La figura precedente è un esempio di albero di ricerca. Il nodo radice è il nodo più in alto. Ogni nodo è costituito da un vettore ordinato di indici, e tra un indice e il suo successivo c'è il puntatore al nodo figlio. Il nodo foglia ha solo puntatori ai dati. Per cercare il record con indice 72, si parte dal nodo radice, quindi utilizzando il vettori di elementi ordinati, si scende al nodo figlio (indici da 32 al 78). In questo caso il nodo trovato corrisponde ad un nodo foglia, da cui si trova l'indice da trovare e il puntatore al record.

Contatore o chiavi primarie multiple?

Spesso ci si trova a dover creare una tabella con chiave multipla, ma conviene lasciare questa impostazione, o utilizzare un contatore e impostare opportunamente i campi come indici? Ovviamente nel secondo caso, se occorre, è possibile inserire anche un vincolo di univocità tra più campi.
Vediamo come esempio due tabelle correlate, e verifichiamo pregi e difetti dei due metodi. L'esempio in esame sarà relativo all'archiviazione di un documento fattura con intestazione e dettaglio.
Schema E-R



Le tabelle corrispondenti sono: documento (anno, numero_documento, id_gruppo, …) e, documento_dettaglio (anno, numero_documento, id_gruppo, numero_riga, …..).
Applicando un contatore, le tabelle diventano: documento (id, anno, numero_documento, id_gruppo, …), documento_dettaglio (id, id_documento, numero_riga, …).
Notiamo subito il grosso vantaggio del primo caso, ossia la garanzia dell'univocità del record tramite anno, numero_documento e id_gruppo. Tuttavia, nel secondo caso esistono altri vantaggi, di cui alcuni meno ovvi:

1) Assenza di dati ridondanti tra tabelle relazionate;
2) Semplificazioni del codice SQL nelle query join;
3) Semplificazione del passaggio dei parametri nei link o nei form;
4) Possibilità di gestire casi particolari (in questo esempio il caso della fattura bis);
5) Semplificazioni del codice SQL e nel codice PHP per modifiche strutturali (in questo esempio, la gestione di multi-aziende).

Vediamo in dettaglio i punti espressi.
Primo punto. Nel primo metodo la ridondanza è ovvia, in quanto la tabella documento_dettaglio ripete gli stessi campi presenti nella chiave della tabella documento.

Secondo punto.Ipotizziamo di dover eseguire una query collegando le due tabelle tramite LEFT JOIN; nel primo caso, il codice SQL ha una forma simile alla seguente

```
SELECT …
FROM documento AS o
LEFT JOIN documento_dettaglio AS d
  ON d.anno=o.anno
  AND d.id_gruppo=o.id_gruppo
  AND d.numero=o.numero
```

Nel secondo caso avrebbe

```
SELECT …
FROM documento AS o
LEFT JOIN documento_dettaglio AS d ON d.id_documento=o.id
```

In una singola query si nota solo una piccola differenza. Nel primo caso, se consideriamo che all'interno di un gestionale, ci dovrebbero essere tante query come queste, (specie se si applica lo stesso ragionamento anche alle altre tabelle) è molto facile imbattersi in questo errore

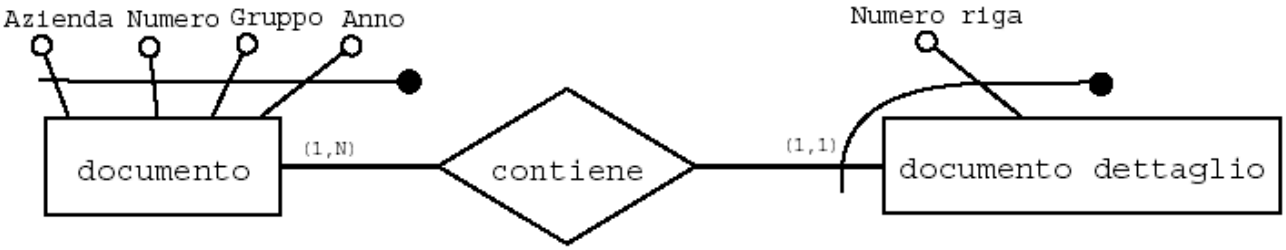
```
SELECT …
FROM documento AS o
LEFT JOIN documento_dettaglio AS d
  AND d.id_gruppo=o.id_gruppo
  AND d.numero=o.numero
```

Se il beta-testing non riesca a rilevare l'errore, esso si scoprirà solo dopo un anno di utilizzo del gestionale.

Terzo punto. Riflettiamo su tutti i link e form dove occorre passare il riferimento di una fattura: nel primo caso dobbiamo passare tre variabili e nel primo basta una sola variabile. Se invece dobbiamo passare il riferimento di una riga della fattura, allora nel primo caso occorrono quattro variabili e nel secondo al massimo due (l'identificativo della fattura e quello della riga).

Quarto punto. Un caso particolare di fattura, è la fattura bis. Questo tipo di fattura si applica quando si ha la necessità di inserire una fattura in un determinato periodo nel passato. Poiché le fatture sono progressive, l'unico modo è quello di duplicare un numero di una fattura, inserendo nella seconda affianco al numero la dicitura bis. È ovvio che solo applicando il secondo metodo, si può gestire un caso così particolare.

Quinto punto. Ipotizziamo di voler convertire il programma di fatturazione in un programma di fatturazione multi-azienda. Quando ci si imbatte in modifiche di questo tipo, occorre rivedere i requisiti del sistema e modificare lo schema E-R. Vediamo di seguito lo schema modificato



Si noti che mentre nel primo metodo, l'identificativo dell' azienda diviene parte della chiave del documento, nel secondo caso diviene un semplice campo in più. Questo vuole dire, che nel primo metodo, il codice SQL per una query che collega le due tabelle diventa

```
SELECT …
FROM documento AS o
LEFT JOIN documento_dettaglio AS d
  ON d.id_azienda=o.id_azienda
  AND d.anno=o.anno
  AND d.id_gruppo=o.id_gruppo
  AND d.numero=o.numero
```

Quindi in questo caso occorre rivedere tutte le query dell'applicativo.
Nel secondo metodo la query rimane invariata, (ovviamente in tutte le query, occorre aggiungere nella clausola WHERE il filtro per id_ azienda); questo si traduce in un minore costo di implementazione.
Si presuppone che l'utente possa visualizzare solo un'azienda, quindi la variabile id_ azienda viene inserita come variabile di sessione. Di conseguenza in entrambi i metodi non occorre modificare link e form.

Chiavi e codici alfanumeri

Vorrei aggiungere un altro tipo di problema che si manifesta con la scelta delle chiavi primarie. Consideriamo la tabella relativa all'anagrafica dei clienti; se impostiamo come chiave primaria la partita IVA, allora vuol dire escludere clienti da nazioni come la Svizzera.. Proprio per evitare questi problemi, molti gestionali impostano delle chiavi alfanumeriche, per clienti, fornitori e articoli. A questo punto sembra di aver risolto tutti i problemi, ma non è così. Questi codici alfanumeri, sono molto utilizzati dagli utenti del sistema come valori di riferimento, e come ogni elemento del sistema, l'utente richiede espressamente di poterli modificare a piacere. Il problema è che per un utente, un codice alfanumerico è una chiave "intellettuale" utile per gli utenti del sistema, e non una chiave "fisica" necessaria per il sistema. Ad esempio ci sono utenti che eliminano vecchi articoli o vecchi clienti e riutilizzano il codice (in realtà un cliente o un articolo associati a delle operazioni, non dovrebbe essere eliminato, ma solo sospeso). Un altro caso è quando un vecchio codice alfanumerico ritenuto non conforme, viene sostituito da un nuovo codice.
Si rende quindi necessario creare un contatore nascosto all'utente come chiave primaria, quindi per l'utente si crea un campo specifico alfanumerico (indicizzato) da utilizzare per richiamare i record. In questo modo, la modifica di questo campo, non crea nessun problema. Ovviamente in fase di modifica e creazione, occorre sempre verificare l'univocità del codice modificato dall'utente.

Operatore JOIN

Chi non conosce l'operatore JOIN, è solito risolvere in questo modo

```
$sql = " SELECT * FROM rubrica ORDER BY  cognome, nome ";
$res = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_array($res))
{
        $id_ provincia = $row['id_ provincia'];
        $sql=" SELECT * FROM provincie WHERE id='$id_provincia' ";
        $res_com = mysql_query($sql);
        if ($row_com = mysql_fetch_array($res_com))
        {
                $provincia = $row_com['nome'];
        }
        ……
        ……
        ……
}
```

Quando si deve lavorare con più tabelle, in molti casi, occorre utilizzare il JOIN che semplifica il codice e riduce in modo drastico il numero delle query.

```
$sql = " SELECT r.*, p.nome AS provincia
          FROM rubrica AS r
          LEFT JOIN provincie AS p ON p.id = r.id_provincia
       ORDER BY r.cognome, r.nome  ";
$res = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_array($res))
{
        ……
        ……
        ……
}
```

Nel precedente esempio sono stati utilizzati degli **alias**: il loro uso è importante, in quanto permettono di distinguere i campi in caso di ambiguità (nomi di campi uguali su tabelle diverse), e migliorano la leggibilità del codice. Se potete, evitate di scrivere la query in questo modo

```
SELECT rubrica.*, provincie.nome AS provincia
FROM rubrica
LEFT JOIN provincie ON provincie.id = id_provincia
ORDER BY cognome, rubrica.nome
```

Si noti che i nomi dei campi cognome e id_provincia poiché appartengono solo alla tabella rubrica, sono distinguibili dal gestore di database, ma sono meno leggibili dall'utente.

Visualizzare il risultato di una query

Per semplificare il codice in PHP necessario per la stampa dei risultati delle query di esempio, è possibile costruire una funzione riutilizzabile. È da notare che la funzione che vi verrà proposta, ha troppe limitazioni, e difficilmente può essere utilizzato in un reale gestionale. Il suo unico utilizzo è quello di visualizzare il risultato di una query tramite una tabella. Di seguito il codice.

```
function stampa_tabella($sql)
{
   $res = mysql_query($sql) or errore_db($sql);
   $campi = mysql_num_fields($res);
```

```
    echo "<table border=\"1\" cellpadding=\"5\" cellspacing=\"0\"><tr>";
    for ($i=0; $i<$campi; $i++)
    {
      echo "<td><b>".mysql_field_name($res, $i)."</b></td>";
    }
    echo "</tr>";
    while ($row = mysql_fetch_row($res))
    {
      echo "<tr>";
      for ($i=0; $i<$campi; $i++)
      {
         $valore = htmlspecialchars($row[$i]);
         if ($valore == '') $valore=" ";
         echo "<td>$valore</td>";
      }
      echo "</tr>";
    }
    echo "</table>";
}
```

Tipi diversi di JOIN

In SQL esistono tre tipi di operatore JOIN

- INNER JOIN
- LEFT JOIN
- RIGHT JOIN

Vediamo alcuni esempi per capire le differenze. Prendiamo in esame le seguenti tabelle

Tabella impiegati

id	**nome**	**id_reparto**
1	Andrea	3
2	Filippo	2
3	Mauro	NULL
4	Piero	2
5	Paolo	1
6	Roberto	2

Tabella reparti

id	**nome**
1	Amministrazione
2	Magazzino
3	Ufficio Acquisti
4	Produzione

INNER JOIN

```
SELECT i.nome AS Impiegato, r.nome AS Reparto
FROM impiegati AS i
INNER JOIN reparti AS r ON r.id=i.id_reparto
```

Impiegato	Reparto
Paolo	Amministrazione
Filippo	Magazzino
Piero	Magazzino
Roberto	Magazzino
Andrea	Ufficio Acquisti

La query restituisce le righe delle tabelle solo dove c'è un legame tra r.id e i.id_reparto

LEFT JOIN

```
SELECT i.nome AS Impiegato, r.nome AS Reparto
FROM impiegati AS i
LEFT JOIN reparti AS r ON r.id=i.id_reparto
```

Impiegato	Reparto
Andrea	Ufficio Acquisti
Filippo	Magazzino
Mauro	NULL
Piero	Magazzino
Paolo	Amministrazione
Roberto	Magazzino

La query restituisce tutte le righe della tabella impiegato, il campo Reparto assume il valore r.nome solo dove c'è un legame tra r.id e i.id_reparto, altrimenti assume NULL.

RIGHT JOIN

```
SELECT i.nome AS Impiegato, r.nome AS Reparto
FROM impiegati AS i
RIGHT JOIN reparti AS r ON r.id=i.id_reparto
```

Impiegato	Reparto
Paolo	Amministrazione
Filippo	Magazzino
Piero	Magazzino
Roberto	Magazzino
Andrea	Ufficio Acquisti
NULL	Produzione

La query restituisce tutte le righe della tabella reparto, il campo Impiegato assume il valore i.nome solo dove c'è un legame tra r.id e i.id_reparto, altrimenti assume NULL.

SubQuery

Una subquery è una query situata all'interno di una interrogazione. Se una query restituisce una tabella, essa può sostituire una tabella in una interrogazione, se invece restituisce un valore allora può sostituire un campo. Per vedere alcuni esempi, prendiamo in esame le seguenti tabelle

Tabella cliente

id	**ragione**	**id_categoria**
1	Fai da te	2
2	Il paradiso della brucola	2
3	XXX srl	1
4	Carrozzeria Ferri snc	1
5	Bar dello sport	3
6	Colore vivo	2

Tabella categoria

id	**nome**
1	Carrozzeria
2	Ferramenta
3	Altro

Tabella articolo

id	**descrizione**	**scorta_fisica**	**scorta_sicurezza**	**id_categoria**
1	Set di chiavi	0	15	2
2	Kit di chiodi assortiti	0	5	1
3	Pinza	2	10	2
4	Punte trapano	4	5	3
5	Lame per seghetti	0	5	3
6	Rivettatrice	0	8	2

Tabella ordine_cliente_dettaglio

id	**id_ordine**	**rigo**	**id_articolo**	**qta_ordinata**	**qta_evasa**	**prezzo**
1	1	1	1	1	1	15
2	1	2	2	2	2	15
3	2	1	3	1	1	15
4	3	1	1	2	2	15
5	4	1	4	1	1	16
6	4	2	5	1	0	20
7	5	1	2	1	1	15
8	5	2	1	2	1	15
9	5	3	6	2	0	12
10	6	1	5	2	0	20
11	7	1	5	1	0	20
12	8	1	4	1	1	16
13	8	2	1	1	0	15
14	9	1	2	2	1	15
15	10	1	6	1	0	12
16	10	2	2	2	1	15

Tabella ordine_cliente

id	data_ordine	id_cliente	nr_ordine
1	14/01/2008	1	1
2	14/01/2008	3	2
3	15/01/2008	3	3
4	15/01/2008	4	4
5	16/01/2008	5	5
6	16/01/2008	6	6
7	16/01/2008	1	7
8	17/01/2008	2	8
9	18/01/2008	3	9
10	18/01/2008	4	10

Tabella categoria_merceologica

id	nome
1	Minuterie
2	Utensili
3	Accessori

Tabella ordine_fornitore

id	data_ordine	id_fornitore	nr_ordine
1	14/01/2008	1	1
2	14/01/2008	2	2

Tabella ordine_fornitore_dettaglio

id	id_ordine	Rigo	id_articolo	qta_ordinata	qta_evasa	prezzo
1	1	1	1	10	0	5
2	1	2	2	5	0	7
3	1	3	4	5	0	8
4	1	4	3	5	0	4
5	2	1	5	10	0	8
6	2	2	7	10	0	5
7	2	3	6	10	0	5

Di seguito la query per calcolare il totale importo ordinato per cliente

```
SELECT c.ragione AS Cliente, sum(d.prezzo*d.qta_ordinata) AS Totale
FROM clienti AS c
INNER JOIN ordine_cliente AS o ON o.id_cliente=c.id
INNER JOIN ordine_cliente_dettaglio AS d ON d.id_ordine=o.id
GROUP BY c.id
```

Equivalente a

```
SELECT c.ragione AS Cliente, t.Totale
FROM cliente AS c
INNER JOIN (
  SELECT o.id_cliente, sum(d.prezzo*d.qta_ordinata) AS Totale
  FROM ordine_cliente AS o
  INNER JOIN ordine_cliente_dettaglio AS d ON d.id_ordine=o.id
  GROUP BY o.id_cliente
) AS t ON t.id_cliente=c.id
```

In questo caso, la sotto query sostituisce una tabella nell'interrogazione. La sub-query, calcola il totale importo ordinato per id_cliente.

In entrambi i casi abbiamo il seguente risultato

Cliente	Totale
Fai da te	45
Il paradiso della brucola	20
XXX srl	76
Carrozzeria Ferri snc	66
Bar dello sport	69
Colore vivo	28

Per adesso abbiamo visto una query relativamente semplice e noiosa, ma vediamo cosa succede se si deve calcolare per ogni cliente:

- Totale importo ordinato;
- Totale importo evaso (si intende la valutazione delle merce consegnata);
- Totale importo da evadere (si intende la valutazione delle merce da consegnare);
- Varietà della tipologia di articoli (ovvero quanti articoli diversi);
- Categoria merciologica più richiesta (intesa come quantità di pezzi ordinati);

per semplicità verrà proposta solo la versione con le subquery, inoltre per migliorare la leggibilità del codice verrà proposta la versione implementata con il PHP

```
$sql_importo = "
    SELECT o.id_cliente, sum(d.prezzo*d.qta_ordinata) AS totale_ordinato,
                sum(d.prezzo*d.qta_evasa) AS totale_evaso,
                sum(d.prezzo*( d.qta_ordinata - d.qta_evasa)) AS totale_da_evadere
    FROM ordine_cliente AS o
    INNER JOIN ordine_cliente_dettaglio AS d ON d.id_ordine=o.id
    GROUP BY o.id_cliente ";
$sql_temp = "
    SELECT o2.id_cliente, d2.id_articolo, sum(d2.qta_ordinata) AS tot_qta_ordinata
    FROM ordine_cliente AS o2
    INNER JOIN ordine_cliente_dettaglio AS d2 ON d2.id_ordine=o2.id
    GROUP BY o2.id_cliente, d2.id_articolo ";
$sql_varietà = "
    SELECT v.id_cliente, sum(1) AS Varietà
    FROM ($sql_temp) AS v
    GROUP BY v.id_cliente ";



$sql_categoria = "
    SELECT m.nome
    FROM ($sql_temp) AS v2
    INNER JOIN articolo AS a ON a.id = v2.id_articolo
    INNER JOIN categoria_merciologica AS m ON m.id = a.id_categoria
    WHERE v2.id_cliente=c.id
    ORDER BY v2.tot_qta_ordinata DESC
    LIMIT 1 ";

$sql = "
    SELECT c.ragione AS Cliente, i.totale_ordinato AS `Totale Ordinato`,
                i.totale_evaso AS `Totale Evaso`,
```

```
        i.totale_da_evadere AS `Totale da Evadere`,
        r.Varietà,
        ($sql_categoria) AS Categoria
FROM cliente AS c
INNER JOIN ($sql_importo) AS i ON i.id_cliente=c.id
INNER JOIN ($sql_varietà) AS r ON r.id_cliente=c.id ";
```

Si noti che la query ($sql_categoria) restituisce un valore, e quindi può andare a sostituire un campo nella SELECT.
Di seguito il risultato della query

Cliente	Ordinato	Evaso	Da evadere	Varietà	Categoria
Fai da te	45	45	0	2	Minuterie
Il paradiso della brucola	20	0	20	1	Accessori
XXX srl	76	61	15	3	Utensili
Carrozzeria Ferri snc	66			3	Minuterie
Bar dello sport	69	30	39	3	Utensili
Colore vivo	28			1	Accessori

Transazioni protette

Una transazione si intende un insieme di una o più interrogazioni a database. Prima di spiegare cosa sia una transazione protetta, è necessario mostrare alcuni problemi che possono sorgere in un gestionale con normali transazioni. Prendiamo come esempio un semplice programma che gestisce un magazzino, gli ordini dei clienti e gli ordini dei fornitori. In particolare deve avere le seguenti funzionalità:

- Visualizzazione delle disponibilità di magazzino;
- Scarico del magazzino mediante evasione degli ordini dei clienti;
- Carico del magazzino mediante evasione degli ordini dei fornitori;
- Visualizzazione dei articoli da ordinare a fornitore;
- Esportazione delle disponibilità verso gli agenti fuori sede, e verso i clienti;

Il programma riprende fedelmente le tabelle proposte negli esempi precedenti.

Eseguire lo scarico del magazzino mediante evasione degli ordini dei clienti, vuol dire essenzialmente eseguire due query. Vediamo il codice relativo allo scarico dell'articolo $id_articolo di quantità $qta, relativo al dettaglio ordine $id_dettaglio_ordine

```
$sql = " UPDATE ordine_cliente_dettaglio
        SET qta_evasa = (qta_evasa + $qta)
        WHERE id = '$id_dettaglio_ordine' ";
mysql_query($sql) or errore_db($sql);

$sql = " UPDATE articolo
        SET scorta_fisica = (scorta_fisica - $qta)
        WHERE id = '$id_articolo' ";
mysql_query($sql) or errore_db($sql);
```

Poiché l'operazione viene eseguita in due query, ed il gestore di database lavora mediante dei processi, può accadere che tra la fine della prima esecuzione e l'inizio della seconda, ci siano le esecuzioni di altre query in corso. Una visualizzazione delle disponibilità potrebbe far risultare che l'articolo $id_articolo, ha una giacenza maggiore di quella reale. Lo stesso problema si ha con l'esportazione delle disponibilità, o con la visualizzazione degli articoli da ordinare a fornitore. La probabilità che questo avvenga dipende dal numero degli utenti che lavorano con il gestionale e dal numero delle operazioni che essi svolgono, e di solito è relativamente bassa. In oltre tutte le operazioni successive, tornano ad essere corrette.
Un altro tipo di problema, ancora più grave, si potrebbe verificare quando, all'interno di un certo numero di query "critiche" avviene un blocco del programma, ad esempio per via di un errore. In questo caso poiché l'operazione non verrà mai completata, le anomalie rimangono presenti nel sistema fino a quando non si effettua un intervento correttivo. La probabilità che avvenga un blocco all'interno di una operazione delicata, dipende da quanti errori possono esserci nel programma, e solitamente (in una versione di rilascio) è molto bassa.
Anche se ci sono poche probabilità di avere dei problemi è molto meglio proteggere le query tramite transazioni protette. Queste ultime consentono di gestire un certo numero di query come se fossero una sola. L'inizio di una transazione protetta avviene tramite l'istruzione SQL "START TRANSACTION" e finisce con l'istruzione "COMMIT". L'effetto delle query all'interno di queste istruzioni, si hanno solo quando si è eseguita la "COMMIT", quindi un'interrogazione eseguita contemporaneamente a questa non potrà vedere le modifiche apportate. Se il programma si blocca, tutte le query eseguite successivamente a "START TRANSACTION", sono automaticamente annullate. Lo stesso effetto si ha eseguendo l'istruzione SQL "ROLLBACK". Quindi possiamo riscrivere il codice in questo modo

```
mysql_query(" START TRANSACTION ") or errore_db(" START TRANSACTION ");

$sql = " UPDATE ordine_cliente_dettaglio
        SET qta_evasa = (qta_evasa + $qta)
```

```
        WHERE id = '$id_dettaglio_ordine' ";
mysql_query($sql) or errore_db($sql);

$sql = " UPDATE articolo
        SET scorta_fisica = (scorta_fisica - $qta)
        WHERE id = '$id_articolo' ";
mysql_query($sql) or errore_db($sql);

mysql_query(" COMMIT ") or errore_db(" COMMIT ");
```

Si osservi che la funzione errore_db(), in caso di errore, blocca l'esecuzione del programma, e quindi se ci sono blocchi all'interno di una transazione, automaticamente le query in esso eseguiti verrebbe annullati.

Ottimizzare le interrogazioni

Un programma lento può dipendere da diversi fattori, dall'hardware del server, dalla tipo di connessione al server, dalla configurazione del gestore di database, dal volume dei dati o anche dalla complessità delle interrogazioni, e per ultimo dalla struttura del database. Quest'ultimo fattore, è poco conosciuto dai principianti, ed quindi quello che verrà discusso tra poco.
Consideriamo l'esempio del gestionale visto precedentemente, ed effettuiamo un'analisi sulle prestazioni nella schermata di visualizzazione degli ordini. Questa schermata viene utilizzata dall'utente per due obbiettivi:

- Visualizzare gli ordini dei clienti con totale importo ordinato e da evadere, nell'arco di un mese;
- Cercare un ordine di un cliente per visionarlo, modificarlo o evaderlo;

La ricerca di un ordine avviene entrando nella schermata della visualizzazione degli ordini, quindi consideriamo questa operazione simile alla visualizzazione, e di conseguenza vanno considerate come unica operazione.
Ciò che si vuole determinare, è se conviene avere due campi ridondanti nella tabella ordini_clienti, rispettivamente totale_importo e totale_daevadere.
Si definisce il volume di una tabella, la quantità di record coinvolti nelle operazioni, nell'arco di un anno.
Vediamo l'anali dei volumi e delle operazioni.

Tabella dei volumi

Tabella	Volume
cliente	60
ordine_cliente	150
ordine_cliente_dettaglio	350

Tabella delle operazioni

Operazione	Frequenza
Inserimento rigo ordine	350 all'anno
Visualiz. ordini	350 all'anno
Evasione di un ordine	200 all'anno

A questo punto possiamo calcolare gli accessi nelle tabelle in base alle due ipotesi: la prima con i campi totale_importo e totale_daevadere nella tabella ordini_clienti, e la seconda senza questi campi.
Di seguito gli accessi per la prima ipotesi

Inserimento rigo ordine

Tabella	Accesso	Tipo
ordine_cliente_dettaglio	1	S
ordine_cliente_dettaglio	2	L
ordine_cliente	1	S

Visualizzazione ordini

Tabella	Accesso	Tipo
cliente	12	L
ordine_cliente	12	L

Evasione di un ordine

Tabella	Accesso	Tipo
ordine_cliente_dettaglio	2	S
ordine_cliente_dettaglio	2	L
ordine_cliente	1	S
articolo	2	S

Si noti che quando si inserisce un rigo d'ordine, o quando lo si evade, occorre aggiornare anche i due campi ridondanti della tabella ordine_cliente, e per farlo occorre rileggere tutti i righi dell'ordine.

Di seguito gli accessi per la seconda ipotesi

Inserimento rigo ordine

Tabella	Accesso	Tipo
ordine_cliente_dettaglio	1	S

Visualizzazione ordini

Tabella	Accesso	Tipo
cliente	12	L
ordine_cliente	12	L
ordine_cliente_dettaglio	29	L

Evasione di un ordine

Tabella	Accesso	Tipo
ordine_cliente_dettaglio	2	S
Articolo	2	S

Si noti che visualizzare un ordine, con relativo totale, vuol dire dover leggere anche tutti i suoi righi. Se consideriamo che un accesso in scrittura, ha un costo doppio di un accesso in lettura, possiamo calcolare il costo per anno delle due ipotesi.

Prima ipotesi

Operazione	Tabella	Accesso	Tipo	Frequenza	Costo
Inserimento rigo ordine	ordine_cliente_dettaglio	1	S	350	700
	ordine_cliente_dettaglio	2	L		700
	ordine_cliente	1	S		700
Visualizzazione ordini	cliente	12	L	350	4200
	ordine_cliente	12	L		4200
Evasione di un ordine	ordine_cliente_dettaglio	2	S	200	800
	ordine_cliente_dettaglio	2	L		400
	ordine_cliente	1	S		400
	articolo	2	S		800
Totale					12900

Seconda ipotesi

Operazione	Tabella	Accesso	Tipo	Frequenza	Costo
Inserimento rigo ordine	ordine_cliente_dettaglio	1	S	350	700
Visualizzazione ordini	cliente	12	L	350	4200
	ordine_cliente	12	L		4200
	ordine_cliente_dettaglio	29	L		10150
Evasione di un ordine	ordine_cliente_dettaglio	2	S	200	800
	ordine_cliente	1	S		400
Totale					20450

Dai risultati si evince che nella visualizzazione degli ordini, il costo del calcolo a run-time del totale degli importi è molto più alto del costo del mantenimento dei due campi da aggiungere alla tabella.
Volendo è possibile determinare anche il costo di memoria aggiuntivo del database per mantenere i due campi. Assumendo che per memorizzare un DOUBLE ci vogliano 8 byte abbiamo 150x8x2=2400 byte cioè circa 2 Kb.
Questo risultato dimostra l'importanza della struttura del database nell'implementazione del programma, e che soprattutto esse non deve essere vista solamente come un modo per archiviare i dati ma tutt'altro deve essere elaborata e concepita per far funzionare meglio il programma.

Programmazione Standard

Questo capitolo illustra alcune tecniche standard di programmazione. La maggior parte degli esempi si basa sull'implementazione di una rubrica. Per ogni problematica, viene dapprima proposto un semplice esempio, e successivamente vengono proposti dei miglioramenti fino ad arrivare ad un sorgente completo. In alcuni casi possono esserci degli approfondimenti, ed in altri casi vengono proposti altri esempi dimostrativi.
In particolare i moduli da sviluppare dovranno implementare la visualizzazione, l'inserimento, la modifica e l'eliminazione dei contatti. Ci deve essere in oltre la possibilità di filtrare e ordinare i dati.
Di seguito la struttura dati

```
CREATE TABLE `rubrica` (
  `id` int(11) NOT NULL auto_increment,
  `nome` varchar(255) default NULL,
  `cognome` varchar(255) default NULL,
  `indirizzo` varchar(255) default NULL,
  `cap` varchar(10) default NULL,
  `comune` varchar(255) default NULL,
  `id_provincia` int(11) default '0',
  `telefono` varchar(255) default NULL,
  `cellulare` varchar(255) default NULL,
  `note` text,
  `email` varchar(255) default NULL,
  PRIMARY KEY  (`id`)
)

CREATE TABLE `provincie` (
  `id` int(11) NOT NULL auto_increment,
  `nome` varchar(255) default NULL,
  `codice` varchar(2) default NULL,
  PRIMARY KEY  (`id`)
)
```

Menu statici in HTML

Menu orizzontali

Vediamo adesso come implementare un semplice menu sulla parte superiore della pagina. Ovviamente, come è stato detto precedentemente, poiché il menu deve comparire in tute schermate, potrebbe essere conveniente implementarlo nel file top.php. Quando occorre implementare schermate senza menu come ad esempio le finestre popup, ci sono due rimedi:

- Creare un file menu.php, staccato quindi da top.php
- Creare un'altro file top_popup.php per le finestre popup

Vediamo nell'esempio due semplici link: la visualizzazione della lista delle persone e inserimento di uno nuovo.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Titolo del sito</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script type="text/javascript" src="<?=url_principale?>javascript/script.js"></script>
<link rel="stylesheet" href="<?=url_principale?>img/stile.css" type="text/css">
</head>
<body style="margin:0">
<table>
<tr>
    <td><a href="index.php">Lista contatti</a></td>
    <td><a href="modifica_contatto.php">Nuovo contatto</a></td>
</tr>
</table>
```

Nel successivo esempio abbiamo aggiunto per ogni link un bottone

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<html>
<head>
<title>Titolo del sito</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
<script type="text/javascript" src="<?=url_principale?>javascript/script.js"></script>
<link rel="stylesheet" href="<?=url_principale?>img/stile.css" type="text/css">
</head>
<body style="margin:0">
<table>
<tr>
    <td align="center"><a href="index.php"><img witdh="32" height="32"
                            src="<?=url_principale?>img/lista_conttati.gif" border="0"
                            title="Visualizza la lista dei contatti"><br>Lista<br>Contatti</a></td>
    <td align="center"><a href="modifica_contatto.php"><img witdh="32" height="32"
                             src="<?=url_principale?>img/nuovo_conttato.gif" border="0"
                             title="Crea un nuovo contatto"><br>Nuovo<br>Contatto</a></td>
</tr>
</table>
```

Si noti l'inserimento dei tooltip: passando sopra le immagini viene visualizzata la descrizione della funzionalità del bottone. Per evitare il bordo dell'immagine, abbiamo impostato l'attributo border a zero.

Nel prossimo esempio vedremo come aggiungere l'effetto rollover alle immagini. Questo particolare tipo di effetto consiste nel cambiare l'immagine al passaggio del mouse, e si ottiene mediante la gestione degli eventi del mouse: dentro (mouseover) e fuori (mouseout). Vediamo il codice

```
<table width="150">
<tr align="center">
     <td align="center"><a href="index.php"><img witdh="32" height="32"
               src="<?=url_principale?>img/lista_conttati.gif" border="0"
               onmouseover="javascript: this.src='<?=url_principale?>img/lista_conttati_over.gif'"
               onmouseout=" javascript: this.src='<?=url_principale?>img/lista_conttati.gif'"
               title="Visualizza la lista dei contatti"><br>Lista<br>Contatti</a></td>
     <td align="center"><a href="modifica_contatto.php"><img witdh="32" height="32"
               src="<?=url_principale?>img/nuovo_conttato.gif" border="0"
               onmouseover="javascript: this.src='<?=url_principale?>img/nuovo_conttato_over.gif'"
               onmouseout=" javascript: this.src='<?=url_principale?>img/nuovo_conttato.gif'"
              title="Crea un nuovo contatto"><br>Nuovo<br>Contatto</a></td>
</tr>
</table>
```

Classe menu_orizzontale

Per agevolare la programmazione, è possibile implementare una classe che implementa i menu orizzontali. Ci sono almeno due vantaggi per implementarlo, la prima è che per implementare un menu con questa classe bastano poche righe di codice, e la seconda è che se occorre migliorare il layout di tutti i menu implementati in un applicativo, basta modificare solo questa classe.

Per l'implementazione di questa classe si adotteranno delle semplici convenzioni:

- Tutte le immagini risiedono tutte sullo stesso percorso, hanno la stessa estensione, la stessa dimensione e sono quadrate;
- Tutte le immagini hanno il rollover, e la seconda immagine ha il suffisso '_over';

Di seguito l'implementazione di un esempio di una classe che costruisce un menu orizzontale

```
class menu_orizzontale
{
  protected $lista_bottoni;
  public $larghezza_immagine;
  public $larghezza_bottone;
  public $percorso;
  public $estensione;

  public function __construct()
  {
     $this->lista_bottoni = array();
     $this->larghezza_immagine = 32;
     $this->larghezza_bottone = 60;
     $this->percorso = url_principale."img/";
     $this->estensione = '.gif';
  }

  public function bottone($img, $testo, $href, $target, $tooltip)
  {
     $btn['img'] = $img;
     $btn['testo'] = $testo;
     $btn['href'] = $href;
     $btn['target'] = $target;
     $btn['tooltip'] = $tooltip;
     $this->lista_bottoni[] = $btn;
```

```
 }


 public function scrivi_html()
 {
   $width = count($this->lista_bottoni)*$this->larghezza_bottone;
    $risultato = '<table cellpadding="0" cellspacing="0" width="'.$width.'"><tr>';
    if (is_array($this->lista_bottoni))
    {
      foreach ($this->lista_bottoni AS $btn)
      {
            $img = $btn['img'];
            $testo = $btn['testo'];
            $href = $btn['href'];
            $target = $btn['target'];
            $tooltip = $btn['tooltip'];
            if ($target) $target=" target=\"$target\" ";
            $risultato .= '<td width="'.$this->larghezza_bottone.'" align="center"><a
                  href="'.$href.'" '.$target.'><img
                  width="'.$this->larghezza_immagine.'" height="'.$this->larghezza_immagine.'"
                  border="0" src="'.$this->percorso.$img.$this->estensione.'"
                  onmouseover="javascript: this.src=\''.$this->percorso.$img.'_over'.$this->estensione.'\'"
                  onmouseout="javascript: this.src=\''.$this->percorso.$img.$this->estensione.'\'"
                  title="'.$tooltip.'"><br>'.$testo.'</a></td>';
      }
   }
   $risultato .= '</tr></table>';

   return $risultato;
 }

}
```

A questo punto possiamo vedere il codice per implementare un menu mediante questa classe

```
$menu = new menu_orizzontale();
$menu->bottone('lista_contatti', 'Lista<br>Contatti', 'index.php', '', 'Visualizza la lista dei contatti');
$menu->bottone('nuovo_contatto', 'Nuovo<br>Contatto', "javascript: ModificaContatto('')", '',
                    'Crea un nuovo contatto');
echo $menu->scrivi_html();
```

Menu verticali

Un menu sulla parte superiore della pagina ha un limite sul numero di link che si possono gestire, mentre un menu sulla parte sinistra della pagina, vincola il resto della pagina ad uno spazio inferiore, che in alcuni casi può diventare veramente un problema. Tramite i fogli di stile è possibile creare un semplice menu senza utilizzare i tag per le liste.
Vediamo un esempio

```
……
……
<style type="text/css">
td.menu{
font-family:Verdana;
```

```
font-size:13px;
font-weight : bold;
text-indent:2px;
text-align:center;
height:30;
}

td.capitolo{
font-family:Verdana;
font-size:12px;
font-weight : bold;
text-indent:2px;
height:25;
vertical-align:bottom;
}

td.paragrafo{
font-family:Verdana;
font-size:12px;
font-weight : normal;
text-indent:20px;
height:18;
}

</style>
<table width="200" cellpadding="0" cellspacing="0" border="0">
<tr>
   <td height="500" width="1" bgcolor="#1010ff"></td>
   <td width="1" bgcolor="#aaaaee"></td>
   <td width="195" bgcolor="#eeeeff" valign="top">
   <table width="100%" cellpadding="0" cellspacing="0" border="0">
   <tr><td class="menu">MENU</td></tr>
   <tr><td class="capitolo">Capitolo 1</td></tr>
   <tr><td class="paragrafo"><a href="paragrafo1_1.html">Paragrafo 1</a></td></tr>
   <tr><td class="paragrafo"><a href="paragrafo1_2.html">Paragrafo 2</a></td></tr>
   <tr><td class="paragrafo"><a href="paragrafo1_3.html">Paragrafo 3</a></td></tr>
   <tr><td class="paragrafo"><a href="paragrafo1_4.html">Paragrafo 4</a></td></tr>
   <tr><td class="capitolo">Capitolo 2</td></tr>
   <tr><td class="paragrafo"><a href="paragrafo2_1.html">Paragrafo 1</a></td></tr>
   <tr><td class="paragrafo"><a href="paragrafo2_2.html">Paragrafo 2</a></td></tr>
   <tr><td class="paragrafo"><a href="paragrafo2_3.html">Paragrafo 3</a></td></tr>
   <tr><td class="paragrafo"><a href="paragrafo2_4.html">Paragrafo 4</a></td></tr>
   </table>
   </td>
   <td width="1" bgcolor="#aaaaee"></td>
   <td width="1" bgcolor="#1010ff"></td>
   <td width="1" bgcolor="#aaaaaa"></td>
</tr>
</table>
```

Utilizzando text-indent, il testo del paragrafo viene spostato di 20px più a destra, mentre il capitolo viene spostato di soli 2px. Affianco ai link possono essere inseriti anche delle immagini, a rappresentare l'argomento del paragrafo.

Come per i menu orizzontali, anche nei menu verticali è possibile inserire delle immagini con tooltip ed anche l'effetto rollover.

Classe menu_verticale

Come per i menu orizzontali per agevolare la programmazione, è possibile implementare una classe che implementa i menu verticali. Tutte e due le classi hanno gli stessi principi di base.

Di seguito l'implementazione di un esempio di una classe che costruisce un menu verticale

```
class menu_verticale
{
  protected $titolo;
  protected $lista_bottoni;
  protected $width;
  public $larghezza_immagine;
  public $percorso;
  public $estensione;
  public $classe_menu;
  public $classe_gruppo;
  public $classe_bottone;

  public function __construct($titolo='', $width='100%')
  {
     $this->lista_bottoni = array();
     $this->larghezza_immagine = 16;
     $this->percorso = url_principale."img/";
     $this->estensione = '.gif';
     $this->width = $width;
     $this->titolo = $titolo;
     $this->classe_menu = 'menu';
     $this->classe_gruppo = 'menu_gruppo';
     $this->classe_bottone = 'menu_bottone';
  }

  public function gruppo($gruppo)
  {
     $btn['gruppo'] = $gruppo;
     $this->lista_bottoni[] = $btn;
  }

  public function bottone($img, $testo, $href, $target, $tooltip)
  {
     $btn['img'] = $img;
     $btn['testo'] = $testo;
     $btn['href'] = $href;
     $btn['target'] = $target;
     $btn['tooltip'] = $tooltip;
     $this->lista_bottoni[] = $btn;
  }

  public function scrivi_html()
  {
     $risultato = '<table cellpadding="0" cellspacing="0" width="'.$this->width.'">';
     if ($this->titolo) $risultato .= '<tr><td class="'.$this->classe_menu.'">'.$this->titolo.'</td></tr>';
```

```
    if (is_array($this->lista_bottoni))
    {
      foreach ($this->lista_bottoni AS $btn)
      {
            $img = $btn['img'];
            $testo = $btn['testo'];
            $href = $btn['href'];
            $target = $btn['target'];
            $tooltip = $btn['tooltip'];
            $gruppo = $btn['gruppo'];
            if ($gruppo)
            {
                $risultato .= '<tr><td class="'.$this->classe_gruppo.'">'.$gruppo.'</td></tr>';
                continue;
            }
            if ($target) $target=" target=\"$target\" ";
            $risultato .= '<tr><td class="'.$this->classe_bottone.'"><a
                        href="'.$href.'" '.$target.'><img
                        width="'.$this->larghezza_immagine.'" height="'.$this->larghezza_immagine.'"
                        border="0" src="'.$this->percorso.$img.$this->estensione.'"
                        onmouseover="javascript: this.src=\''.$this->percorso.$img.'_over'.$this->estensione.'\'"
                        onmouseout="javascript: this.src=\''.$this->percorso.$img.$this->estensione.'\'"
                        title="'.$tooltip.'"></a> <a
                        href="'.$href.'" '.$target.'>'.$testo.'</a></td></tr>';
      }
    }
    $risultato .= '</table>';

    return $risultato;
  }

}
```

Di seguito il codice per implementare un menu mediante questa classe

```
$menu = new menu_verticale('MENU');
$menu->classe_gruppo = 'capitolo';
$menu->classe_bottone = 'paragrafo';
$menu->gruppo('Capitolo 1');
$menu->bottone('pallina', 'Paragrafo 1', ' paragrafo1_1.html', '', 'Paragrafo 1');
$menu->bottone('pallina', 'Paragrafo 2', ' paragrafo1_2.html', '', 'Paragrafo 2');
$menu->bottone('pallina', 'Paragrafo 3', ' paragrafo1_3.html', '', 'Paragrafo 3');
$menu->bottone('pallina', 'Paragrafo 4', ' paragrafo1_4.html', '', 'Paragrafo 4');
$menu->gruppo('Capitolo 2');
$menu->bottone('pallina', 'Paragrafo 1', ' paragrafo2_1.html', '', 'Paragrafo 1');
$menu->bottone('pallina', 'Paragrafo 2', ' paragrafo2_2.html', '', 'Paragrafo 2');
$menu->bottone('pallina', 'Paragrafo 3', ' paragrafo2_3.html', '', 'Paragrafo 3');
$menu->bottone('pallina', 'Paragrafo 4', ' paragrafo2_4.html', '', 'Paragrafo 4');
echo $menu->scrivi_html();
```

Stampa di una tabella a video

Una delle operazioni più comuni in php è quello di prendere dei dati da un database e stamparli in output tramite tabella HTML. Di seguito il codice.

```
<?php
include('include/parametri.php');
include(percorso_principale.'include/connessione.php');
include(percorso_principale.'include/top.php');
?>
<center>
<br>
<table cellspacing="2" cellpadding="5" border="1">
<tr>
        <td><b>Cognome</b></td>
        <td><b>Nome</b></td>
        <td><b>Comune</b></td>
        <td><b>Indirizzo</b></td>
        <td><b>Recapiti Telefonici</b></td>
        <td><b>E-Mail</b></td>
</tr>
<?php
$sql = " SELECT r.*, p.nome AS provincia
         FROM rubrica AS r
         LEFT JOIN provincie AS p ON p.id = r.id_provincia
        ORDER BY r.cognome, r.nome  ";
$res = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_array($res))
{
        $provincia      = htmlspecialchars ($row['provincia']);
        $email          = htmlspecialchars ($row['email']);
        $comune         = htmlspecialchars ($row['cap']." ". $row['comune']);
        if ($provincia) $comune .= " ($provincia)";
?>
<tr>
        <td><?php echo htmlspecialchars($row['cognome']); ?></td>
        <td><?php echo htmlspecialchars($row['nome']); ?></td>
        <td><?php echo $comune; ?></td>
        <td><?php echo htmlspecialchars($row['indirizzo']); ?></td>
        <td><?php echo htmlspecialchars($row['telefono']. " - ".$row['cellulare']); ?></td>
        <td><a href="mailto:<?php echo $email; ?> "><?php echo $email; ?></a></td>
</tr>
<?php
}
?>
</table>
</center>
<?php
include(percorso_principale.'include/bottom.php');
?>
```

Mediante piccoli accorgimenti è possibile aumentare lo spazio a disposizione per i testi contenuti nella tabella. Ad esempio invece di visualizzare l'e-mail mediante un testo, conviene visualizzarlo come tooltip di un'immagine. Di seguito il codice

```
<a href="mailto:<?php echo $email; ?>"><img src="<?=url_principale?>img/email.gif" width="16" height="16"
    border="0" title="Email: <? echo $email; ?>"></a>
```

Per una migliore leggibilità dei dati, solitamente si impostano delle righe alterne colorate.

```
<?php
include('include/parametri.php');
include(percorso_principale.'include/connessione.php');
include(percorso_principale.'include/top.php');
?>
<center>
<br>
<table cellspacing="2" cellpadding="5" border="0">
<tr bgcolor="#AAAAAA">
        <td><b>Cognome</b></td>
        <td><b>Nome</b></td>
        <td><b>Comune</b></td>
        <td><b>Indirizzo</b></td>
        <td><b>Recapiti Telefonici</b></td>
        <td> </td>
</tr>
<?php
$colore1 = "#F0F0FF";
$colore2 = "#E0E0EF";
$colore_corrente = $colore1;
$sql = " SELECT r.*, p.nome AS provincia
          FROM rubrica AS r
          LEFT JOIN provincie AS p ON p.id = r.id_provincia
       ORDER BY r.cognome, r.nome  ";
$res = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_array($res))
{
        $provincia      = htmlspecialchars ($row['provincia']);
        $email          = htmlspecialchars ($row['email']);
        $comune         = htmlspecialchars ($row['cap']." ". $row['comune']);
        if ($provincia) $comune .= " ($provincia)";
?>
<tr bgcolor="<?php echo $colore_corrente; ?>">
        <td><?php echo htmlspecialchars($row['cognome']); ?></td>
        <td><?php echo htmlspecialchars($row['nome']); ?></td>
        <td><?php echo $comune; ?></td>
        <td><?php echo htmlspecialchars($row['indirizzo']); ?></td>
        <td><?php echo htmlspecialchars($row['telefono']. " - ".$row['cellulare']); ?></td>
        <td><a href="mailto:<?php echo $email; ?>"><img src="<?=url_principale?>img/email.gif" width="16"
                height="16" border="0" title="Email: <? echo $email; ?>"></a></td>
</tr>
<?php
        If ( $colore_corrente == $colore1 ) $colore_corrente = $colore2;
        else $colore_corrente = $colore1;
}
?>
```

```
</table>
</center>
<?php
include(percorso_principale.'include/bottom.php');
?>
```

Si noti che se avessimo voluto assegnare lo stesso colore a tutte le righe, avremmo applicato lo stile al tag <TD>, ma poiché vogliamo assegnare colori diversi occorre applicare il colore riga per riga.

Un po' di javascript

Con il javascript possiamo colorare la riga al di sotto del mouse. Utilizzando l'evento *onMouseOver* (Mouse in ingresso) e l'evento *onMouseOut* (Mouse in uscita), si può applicare un colore particolare quando il mouse passa sopra il rigo, e rimettere il colore precedente quando il mouse si sposta in un altro rigo.

```
……
……..
$colore_mouse = "#9FC0F0";
……
……..
<tr bgcolor="<?php echo $colore_corrente; ?>"
    onMouseOver="this.bgColor='<?php echo $colore_mouse; ?>'"
    onMouseOut="this.bgColor='<?php echo $colore_corrente; ?>'">
……
……..
```

Scorrimento della tabella

Quando in una tabella ci sono tanti dati, la pagina diventa più lunga dello schermo, quindi l'utente per visionare i dati, deve utilizzare la barra di scorrimento della pagina. Utilizzando in modo appropriato il tag <DIV> è possibile inserire una barra di scorrimento all'interno tabella, in modo che la pagina rimanga di dimensioni prefissate. Per inserire la barra all'interno di un tag <DIV> si imposta lo stile OVERFLOW ad AUTO, in questo modo quando la parte interna del contenitore eccede le dimensioni prefissate, viene visualizzata una barra di scorrimento. Poiché vogliamo che la prima riga della tabella (contenente l'intestazione) rimanga sempre visibile, occorre creare due tabelle, la prima contenente l'intestazione, e la seconda (contenente i dati) posta all'interno di un contenitore (con lo stile OVERFLOW settato ad AUTO). Ovviamente le larghezze delle colonne delle due tabelle devono essere impostate in modo da allinearsi. La larghezza del contenitore deve tenere conto anche della barra di scorrimento.
Vediamo il codice di seguito

```
<?php
include('include/parametri.php');
include(percorso_principale.'include/connessione.php');
include(percorso_principale.'include/top.php');
?>
<center>
<br>
<div style="border: 1px solid #aaaaaa; width:700px;">
<table width="700" cellspacing="2" cellpadding="5" border="0">
<tr bgcolor="#AAAAAA">
        <td width="100"><b>Cognome</b></td>
        <td width="100"><b>Nome</b></td>
        <td width="150"><b>Comune</b></td>
        <td width="140"><b>Indirizzo</b></td>
        <td width="150"><b>Recapiti Telefonici</b></td>
        <td width="60"> </td>
</tr>
```

```
</table>
<div style="overflow:auto; width:100%; height:300px;">
<table width="680" cellspacing="2" cellpadding="5" border="0">
<?php
$colore_mouse = "#9FC0F0";
$colore1 = "#F0F0FF";
$colore2 = "#E0E0EF";
$colore_corrente = $colore1;
$sql = " SELECT r.*, p.nome AS provincia
          FROM rubrica AS r
          LEFT JOIN provincie AS p ON p.id = r.id_provincia
         ORDER BY r.cognome, r.nome  ";
$res = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_array($res))
{
          $provincia        = htmlspecialchars ($row['provincia']);
          $email            = htmlspecialchars ($row['email']);
          $comune           = htmlspecialchars ($row['cap']." ". $row['comune']);
          if ($provincia) $comune .= " ($provincia)";
?>
<tr bgcolor="<?php echo $colore_corrente; ?>"
    onMouseover="this.bgColor='<?php echo $colore_mouse; ?>'"
    onMouseOut="this.bgColor='<?php echo $colore_corrente; ?>'">
          <td width="100"><?php echo htmlspecialchars($row['cognome']); ?></td>
          <td width="100"><?php echo htmlspecialchars($row['nome']); ?></td>
          <td width="150"><?php echo $comune; ?></td>
          <td width="140"><?php echo htmlspecialchars($row['indirizzo']); ?></td>
          <td width="150"><?php echo htmlspecialchars($row['telefono']. " - ".$row['cellulare']); ?></td>
          <td width="30"><a href="mailto:<?php echo $email; ?>"><img src="<?=url_principale?>img/email.gif"
                              width="16" height="16" border="0" title="Email: <? echo $email; ?>"></a></td>
</tr>
<?php
          If ( $colore_corrente == $colore1 ) $colore_corrente = $colore2;
          else $colore_corrente = $colore1;
}
?>
</table>
</div>
</div>
</center>
<?php
include(percorso_principale.'include/bottom.php');
?>
```

Ricerche e filtri

Prima di far vedere come creare un FORM, occorre soffermarsi un attimo su come si crea una SELECT a partire da una query.
Ecco un semplice caso di scelta di una provincia (scelta obbligatoria)

```
<select name="id_provincia">
<?php
$query = "SELECT id, nome FROM provincie ORDER BY nome";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
    $nome = htmlspecialchars ($row['nome']);
    $id = $row[‘id’];
    echo "\n<option value=\”$id\”>$nome</option>";
}
?>
</select>
```

Se la scelta non deve essere obbligatoria, basta aggiungere appena dopo il tag <SELECT>, la scelta vuota

```
<option value="0"></option>
```

Con le istruzioni precedenti vedremmo la SELECT impostata alla prima opzione, se invece vogliamo una impostazione diversa (memorizzata nella variabile $id_provincia) avremmo come codice il seguente

```
<select name="id_provincia">
<?php
$query = "SELECT id, nome FROM provincie ORDER BY nome";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
    $nome = htmlspecialchars ($row['nome']);
    $id = $row['id'];
    $selected='';
    if ($id==$id_provincia) $selected='selected';
    echo "\n<option $selected value=\"$id\">$nome</option>";
}
?>
</select>
```

Vediamo adesso come implementare un FORM per inserire i parametri di ricerca, e successivamente come visualizzare il risultato della ricerca.
Prenderemo come esempio la tabella rubrica del paragrafo precedente.

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<table width="700" cellspacing="0" cellpadding="0" border="0"
        style="border: 1px solid #aaaaaa;">
<tr>
    <td width="350" valige="top">
        <table width="100%" cellspacing="5" cellpadding="2" border="0">
        <tr>
                <td>Nome</td>
                <td><input name="nome" value="<?php echo $_POST['nome']; ?>"></td>
        </tr>
        <tr>
                <td>Cognome</td>
                <td><input name="cognome" value="<?php echo $_POST['cognome']; ?>"></td>
        </tr>
```

```
        <tr>
                <td>E-Mail</td>
                <td><input name="email" value="<?php echo $_POST['email']; ?>"></td>
        </tr>
        </table>
    </td>
    <td width="350" valige="top">
        <table width="100%" cellspacing="5" cellpadding="2" border="0">
        <tr>
                <td>Comune</td>
                <td><input name="comune" value="<?php echo $_POST['comune']; ?>"></td>
        </tr>
        <tr>
                <td>Provincia</td>
                <td>
       <select name="id_provincia">
        <?php
           $query = "SELECT id, nome FROM provincie ORDER BY nome";
           $res = mysql_query($query) or errore_db($query);
           while ($row = mysql_fetch_array($res))
           {
              $nome = htmlspecialchars ($row['nome']);
              $id = $row['id'];
              $selected='';
              if ($id==$_POST['id_provincia']) $selected='selected';
              echo "\n<option $selected value=\"$id\">$nome</option>";
           }
         ?>
         </select>
                </td>
        </tr>
        <tr>
                <td>Recapiti</td>
                <td><input name="recapiti" value="<?php echo $_POST['recapiti']; ?>"></td>
        </tr>
        </table>
    </td>
</tr>
<tr>
    <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

In un FORM, Si consiglia di impostare la proprietà METHOD sempre a a POST. Si noti che con l'utilizzo di $_SERVER['PHP_SELF'] possiamo facilmente riutilizzare la porzione di codice.

Clausole where

Prima di far vedere come aggiungere dei filtri alla clausola WHERE del codice SQL, colgo l'occasione per spiegare un tipico errore di progettazione (attenzione non di programmazione).
Come esempio prendiamo un classico filtro per Nazione/Regione/Provincia/Comune dove i rispettivi valori sono di tipo numerico (provenienti rispettivamente da quattro SELECT). Cerchiamo di creare una stringa denominata '$ricerca' contenente le clausole del filtro. Ovviamente occorre considerare che i campi della ricerca non sono obbligatori. Di seguito un pessimo codice sorgente:

```
if ( (($Nazione != "0") && ($Nazione != "")) || (($Regione != "0") && ($Regione != "")) || (($Provincia != "0") && ($Provincia != "")) || (($Comune != "0") && ($Comune != "")))
{
	$ricerca = " WHERE ";
}
if (($Nazione != "0") && ($Nazione != ""))
{
	if ($ricerca != " WHERE ")
	{
		$ricerca = $ricerca."AND (B.Nazione = '$Nazione') ";
	}
	else
	{
		$ricerca = $ricerca."(B.Nazione = '$Nazione') ";
	}
}

if (($Regione != "0") && ($Regione != ""))
{
	if ($ricerca != " WHERE ")
	{
		  $ricerca = $ricerca."AND (B.Regione = '$Regione') ";
	}
	else
	{
		$ricerca = $ricerca."(B.Regione = '$Regione') ";
	}
}

if (($Provincia != "0") && ($Provincia != ""))
{
	if ($ricerca != " WHERE ")
	{
		$ricerca = $ricerca."AND (B.Provincia = '$Provincia') ";
	}
	else
	{
		$ricerca = $ricerca."(B.Provincia = '$Provincia') ";
	}
}
if (($Comune != "0") && ($Comune != ""))
{
	if ($ricerca != " WHERE ")
	{
		$ricerca = $ricerca."AND (B. Comune = '$Comune') ";
	}
	else
	{
		$ricerca = $ricerca."(B.Comune = '$Comune') ";
	}
}
```

Ecco come può essere riscritto (in modo semplice ed efficace):

```
$ricerca = " WHERE (1=1) ";
if ($Nazione>0) $ricerca .= " AND (B.Nazione = '$Nazione') ";
if ($Regione>0) $ricerca .= " AND (B.Regione = '$Regione') ";
if ($Provincia>0) $ricerca .= " AND (B.Provincia = '$Provincia') ";
if ($Comune>0) $ricerca .= " AND (B. Comune = '$ Comune') ";
```

Si noti come l'espressione *(1=1)* risolva il problema della prima clausola; tutte le altre clausole sono precedute da AND. Questo esempio serve per far comprendere al lettore che anche se un sorgente funziona, non vuol dire che sia ben progettato.

Vediamo adesso come applicare i filtri alla query relativo all'esempio della rubrica.

```
$sql = " SELECT r.*, p.nome AS provincia
        FROM rubrica AS r
        LEFT JOIN provincie AS p ON p.id = r.id_provincia
        WHERE (1=1) ";

if ($_POST_SQL['nome'])             $sql .= " AND r.nome LIKE '%$_POST_SQL[nome]%' ";
if ($_POST_SQL['cognome'])          $sql .= " AND r.cognome LIKE '%$_POST_SQL[cognome]%' ";
if ($_POST_SQL['email'])            $sql .= " AND r.email LIKE '%$_POST_SQL[email]%' ";
if ($_POST_SQL['comune'])           $sql .= " AND r.comune LIKE '%$_POST_SQL[comune]%' ";
if ($_POST_SQL['id_provincia']>0)   $sql .= " AND r.id_provincia = '$_POST_SQL[id_provincia]' ";
if ($_POST_SQL['recapiti'])         $sql .= " AND (  (r.telefono LIKE '%$_POST_SQL[recapiti]%')
                                                OR (r.cellulare LIKE '%$_POST_SQL[recapiti]%') ) ";
$sql .= " ORDER BY r.cognome, r.nome  ";
```

Si noti che per gestire campi numerici provenienti da SELECT (id_provincia) si utilizza l'operatore '=', mentre per gestire campi testo (nome, cognome, email) si utilizza l'operatore 'LIKE'.

Paginazione di una tabella

Se una tabella contiene molti record, e utile suddividere i record in pagine (in modo da visualizzare una pagina per volta) e di utilizzare i filtri per ridurre il numero di record.
Si noti che ridurre una tabella in pagine, vuol dire aggiungere un vincolo alla query. Questo tipo di vincolo viene impostato mediante l'uso di LIMIT.
Per poter calcolare correttamente il numero delle pagine, occorre calcolare il numero di record totale, quindi suddividerlo per il numero di righe che si voglio visualizzare; quindi occorre effettuare due query, la prima per determinare le righe totali, e la seconda è la porzione di pagina che si intende visualizzare. È consigliabile avere una variabile che memorizzi i filtri che dovranno essere applicati alle due query.

```
$sql_add = "";
if ($_POST_SQL['nome'])              $sql_add .= " AND r.nome LIKE '%$_POST_SQL[nome]%' ";
if ($_POST_SQL['cognome'])           $sql_add .= " AND r.cognome LIKE '%$_POST_SQL[cognome]%' ";
if ($_POST_SQL['email'])             $sql_add .= " AND r.email LIKE '%$_POST_SQL[email]%' ";
if ($_POST_SQL['comune'])            $sql_add .= " AND r.comune LIKE '%$_POST_SQL[comune]%' ";
if ($_POST_SQL['id_provincia']>0)  $sql_add .= " AND r.id_provincia = '$_POST_SQL[id_provincia]' ";
if ($_POST_SQL['recapiti'])           $sql_add .= " AND (  (r.telefono LIKE '%$_POST_SQL[recapiti]%')
                                                    OR (r.cellulare LIKE '%$_POST_SQL[recapiti]%') ) ";
```

Vediamo di seguito la query che calcola il numero di record

```
$nr_record = 0;
$sql = " SELECT sum(1) AS Num
         FROM rubrica AS r
         WHERE (1=1) $sql_add ";
$res = mysql_query($sql) or errore_db($sql);
if ($row = mysql_fetch_array($res)) $nr_record = $row['Num'];
if (!$nr_record) $nr_record=0;
```

Da cui si ottiene il numero di pagine

```
$massimo_righe=50;
$pagine=ceil($nr_record/$massimo_righe);
```

Per semplificare il codice, abbiamo inserito qui, il numero massimo di righe per pagina (**$massimo_righe**), poiché solitamente impostato in un file di configurazione, oppure su database.
Indicando con la variabile ($pagina_corrente), il valore del numero di pagina corrente, vediamo come inserire in un FORM i controlli necessari all'utente per spostarsi da una pagina all'altra. Nel nostro esempio ci appoggeremo al FORM già implementato per i filtri.

```
<select name="pagina_corrente" onChange="javascript:submit();">
<?php
for ($i=1; $i<=$pagine; $i++)
{
    $selected="";
    if ($i==$pagina_corrente) $selected="selected";
    echo "<option $selected  value=\"$i\">Pagina $i</option>";
}
?>
</select>
```

È possibile inserire anche dei bottoni che indicano rispettivamente la pagina precedente e la pagina successiva a quella corrente.

```
<script type="text/javascript">
function SelezionaPagina(pagina)
{
  document.filtri.pagina_corrente.value=pagina;
```

```
  document.filtri.submit();
}
</script>
<?php
if ($pagina_corrente>1)
{
?><a href="javascript:SelezionaPagina(<?php echo $pagina_corrente-1; ?>)">
<img src="<?=url_principale?>img/indietro.gif" width="16" height="16" alt="Pagina precedente" border="0">
</a><?php
}
else
{
?><img src="<?=url_principale?>img/indietro_grigio.gif" width="16" height="16" alt="Pagina precedente"
        border="0"><?php
}

echo " ";

if ($pagina_corrente<$pagine)
{
?><a href="javascript:SelezionaPagina(<?php echo $pagina_corrente+1; ?>)">
<img src="<?=url_principale?>img/avanti.gif" width="16" height="16" alt="Pagina successiva" border="0">
</a><?php
}
else
{
?><img src="<?=url_principale?>img/avanti_grigio.gif" width="16" height="16" alt="Pagina successiva"
        border="0"><?php
}
?>
```

Si noti che sulla prima pagina il bottone indietro è disabilitato, e sull'ultima è disabilitato il bottone avanti.
Vediamo la modifica della query per visualizzare le righe della pagina corrente.

```
$sql = " SELECT r.*, p.nome AS provincia
       FROM rubrica AS r
       LEFT JOIN provincie AS p ON p.id = r.id_provincia
       WHERE (1=1) $sql_add ";
$sql .= " ORDER BY r.cognome, r.nome
        LIMIT ".($pagina_corrente-1).",$massimo_righe ";
```

Ordinamento di una tabella

Finora la query era sempre stata ordinata per cognome e nome del contato, ora vediamo come aggiungere nel FORM anche il tipo di ordinamento.

```
$ordinamento=$_POST['ordinamento'];
if (!$ordinamento) $ordinamento='cognome_nome';

$modo=$_POST_SQL['modo'];
if (!$modo) $modo='ASC';
?>
……..
<select name="ordinamento">
<option <?php if ($ordinamento=="cognome_nome") echo "selected"; ?>
        value="cognome_nome">Cognome, Nome</option>
<option <?php if ($ordinamento=="nome_cognome") echo "selected"; ?>
        value="nome_cognome">Nome, Cognome</option>
<option <?php if ($ordinamento=="comune") echo "selected"; ?>
        value="comune">Comune</option>
</select>
 
Modo <select name="modo">
<option <?php if ($modo== "ASC") echo "selected"; ?> value="ASC">ASC</option>
<option <?php if ($modo== "DESC") echo "selected"; ?> value="DESC">DESC</option>
</select>
```

Quindi vediamo la modifica nella query

```
$sql = " SELECT r.*, p.nome AS provincia
       FROM rubrica AS r
       LEFT JOIN provincie AS p ON p.id = r.id_provincia
       WHERE (1=1) $sql_add ";
if ($ordinamento=="cognome_nome")
{
   $sql .= " ORDER BY r.cognome $modo, r.nome $modo ";
}
if ($ordinamento=="nome_cognome")
{
   $sql .= " ORDER BY r.nome $modo, r.cognome $modo ";
}
if ($ordinamento=="comune")
{
   $sql .= " ORDER BY r.comune $modo ";
}
$sql .= " LIMIT ".($pagina_corrente-1).",$massimo_righe ";
```

Le azioni in una visualizzazione di una tabella

Nel paragrafi precedenti abbiamo visto come implementare una visualizzazione di una tabella, ora vedremo come aggiungere delle azioni.
In particolare quello che occorre fare è aggiungere una colonna alla visualizzazione della tabella, dove per ogni riga, vanno aggiunti due pulsanti, la modifica e l'eliminazione del record. Di seguito il codice

```
<tr bgcolor="<?php echo $colore_corrente; ?>"
    onMouseover="this.bgColor='<?php echo $colore_mouse; ?>'"
    onMouseOut="this.bgColor='<?php echo $colore_corrente; ?>'">
        <td width="100"><?php echo htmlspecialchars($row['cognome']); ?></td>
        <td width="100"><?php echo htmlspecialchars($row['nome']); ?></td>
        <td width="150"><?php echo $comune; ?></td>
        <td width="140"><?php echo htmlspecialchars($row['indirizzo']); ?></td>
        <td width="150"><?php echo htmlspecialchars($row['telefono']. " - ".$row['cellulare']); ?></td>
        <td width="20"><a href="mailto:<?php echo $email; ?>"><img src="<?=url_principale?>img/email.gif"
                          width="16" height="16" border="0" title="Email: <? echo $email; ?>"></a></td>
        <td width="50"><a href="modifica_contatto.php?id=<?=$row['id']?>"><img
                         src="<?=url_principale?>img/modifica.gif" width="16" height="16" border="0"
                         title="Modifica il contatto"></a>
                       <a href="<?php echo $_SERVER['PHP_SELF']."?id_elimina=$row[id]"; ?>"
                         onClick="javascript: return confirm('Sei sicuro di voler eliminare il contatto?')"><img
                         src="<?=url_principale?>img/elimina.gif" width="16" height="16" border="0"
                         title="Elimina il contatto"></a></td>
</tr>
```

Si noti nel bottone di eliminazione la presenza dell'evento 'onClick'. Quest'ultimo provoca la visualizzazione del messaggio di conferma. Inoltre l'eliminazione richiama come pagina, se stessa. Vediamo di seguito l'operazione di eliminazione da aggiungere alla pagina.

```
If ($_GET['id_elimina'])
{
    $sql="DELETE FROM rubrica WHERE id = '$_GET[id_elimina]' ";
    mysql_query($sql) or errore_db($sql);
}
```

Si può notare che queste operazioni diano un problema: se un utente ha eseguito una ricerca mediante l'uso dei filtri, l'esecuzione di un'azione di eliminazione o di modifica, provoca la perdita della ricerca.

Eliminazione di un record

Occorre aggiungere un campo nascosto nel FORM della ricerca chiamato 'id_elimina'

```
<input type="hidden" name="id_elimina" value="">
```

Aggiungere la seguente funzione javascrit

```
<script type="text/javascript">
function EliminaRecord(id)
{
  document.filtri.id_elimina.value=id;
  document.filtri.submit();
}
</script>
```

Modificare il link di eliminazione

```
<a href="javascript: EliminaRecord(<?=$row['id']?>);"
                         onClick=" javascript: return confirm('Sei sicuro di voler eliminare il contatto?')"><img
                         src="<?=url_principale?>img/elimina.gif" width="16" height="16" border="0"
                         title="Elimina il contatto"></a>
```

Infine modificare l'operazione di eliminazione

```
If ($_POST['id_elimina'])
{
    $sql="DELETE FROM rubrica WHERE id = ' $_POST_SQL[id_elimina]' ";
    mysql_query($sql) or errore_db($sql);
}
```

In pratica si fa passare il parametro di eliminazione nel FORM di ricerca, preservando i filtri.

Modifica di un record

Il metodo più semplice consiste nel far aprire una finestra popup dove eseguire la modifica, quindi al momento dell'effettiva operazione di aggiornamento, cercare il FORM della finestra chiamante (se esiste) e far eseguire nuovamente la ricerca.

Occorre aggiungere la funzione javascript che apre una finestra popup

```
<script type="text/javascript">
function ModificaContatto(id)
{
    window.open("modifica_contatto.php?id="+id, "ModificaContatto","toolbar=0, location=0, directories=0,
                status=0, menubar=0, scrollbars=0, resizable=0, width=550, height=300, left=100, top=50");
}
</script>
```

La funzione javascript ModificaContatto apre una finestra popup con dimensione (550X300) senza barre (degli strumenti, menu, ecc…).

Di seguito la modifica al link

```
<a href="javascript: ModificaContatto('<?=$row['id']?>');"><img
                    src="<?=url_principale?>img/modifica.gif" width="16" height="16" border="0"
                    title="Modifica il contatto"></a>
```

Nella finestra popup, successivamente alla query di aggiornamento deve esserci il seguente codice javascript

```
<script type="text/javascript">
if (opener.document.filtri)
{
    opener.document.filtri.submit();
}
self.close();
</script>
```

Inserimento e modifica di un record

Precedente avevamo visto come creare un FORM per effettuare delle ricerche, ora vedremo come utilizzare un FORM per effettuare un inserimento o una modifica.
Nella maggior parte dei casi, mediante opportuni accorgimenti, è possibile utilizzare lo stesso codice per implementare le due operazioni di inserimento e modifica. In particolare se la pagina php viene richiamata con la variabile $id in GET viene eseguita la modifica del record (la chiave del record è ovviamente nella variabile $id); quando la pagina php viene richiamata senza variabili in GET, viene eseguito l'inserimento. Il contenuto della variabile $id determina se si tratta di un nuovo record, o di un record da modificare.
Vediamo di seguito lo scheletro del codice per far vedere come può essere implementato.

```
<?php
include('include/parametri.php');
include(percorso_principale.'include/connessione.php');
include(percorso_principale.'include/top_popup.php');

$_POST_SQL = array_map('post_a_sql', $_POST);
$_GET_SQL = array_map('post_a_sql', $_GET);
$_POST = array_map('post_a_uso', $_POST);
$_GET = array_map('post_a_uso', $_GET);

if ($_POST['azione'])
{
    // recupero delle informazioni contenute nella variabile $_POST
    ……
    ……
    if ($_POST['id'])
    {
        // query di modifica
        ……
        ……
    }
    else
    {
        // query di inserimento
        ……
        ……
    }
     // alla fine dell'operazione di inserimento/modifica aggiornare se possibile la lista dei contatti e chiudersi
?>
<script type="text/javascript">
if (opener.document.filtri)
{
     opener.document.filtri.submit();
}
self.close();
</script>
<?php
     include(percorso_principale.'include/bottom.php');
     exit();
}
……
……
```

```
if ($_GET['id'])
{
    // recupero delle informazioni tramite query del record con chiave $_GET_SQL['id']
    ……
    ……
}
else
{
    // impostazioni delle variabili per inserimento
    ……
    ……
}
……
……
?>
……
……
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="yes">
<input type="hidden" name="id" value="<?php echo $_GET['id']; ?>">
……
……
```

La variabile $azione viene utilizzata per verificare se far apparire il FORM di inserimento/modifica, oppure se fare la query di inserimento/modifica. Nel secondo caso, successivamente alla query viene eseguita la ricerca nella pagina chiamante quindi la finestra viene chiusa.

La query di modifica è implementata di seguito

```
$sql = " UPDATE rubrica
         SET
         nome          = '$_POST_SQL[nome]',
         cognome       = '$_POST_SQL[cognome]',
         indirizzo     = '$_POST_SQL[indirizzo]',
         cap           = '$_POST_SQL[cap]',
         comune        = '$_POST_SQL[comune]',
         id_provincia  = '$_POST_SQL[id_provincia]',
         telefono      = '$_POST_SQL[telefono]',
         cellulare     = '$_POST_SQL[cellulare]',
         email         = '$_POST_SQL[email]'
         WHERE id = '$_POST_SQL[id]' ";
mysql_query($sql) or errore_db($sql);
```

Ecco quella di inserimento

```
$sql = " INSERT INTO rubrica (nome, cognome, indirizzo, cap, commune, id_provincia, telefono,cellulare, email)
         VALUES ('$_POST_SQL[nome]', '$_POST_SQL[cognome]', '$_POST_SQL[indirizzo]',
         '$_POST_SQL[cap]', '$_POST_SQL[comune]', '$_POST_SQL[id_provincia]', '$_POST_SQL[telefono]',
         '$_POST_SQL[cellulare]', '$_POST_SQL[email]') ";
mysql_query($sql) or errore_db($query);
```

Il FORM di inserimento/modifica ha una struttura simile al FORM di ricerca. In dettaglio, la differenza è che i valori non provengono da una ricerca precedente (quindi contenuti in $_POST) ma da un query. In oltre occorre aggiungere due campi nascosti al FORM: uno utilizzato per gestire il flusso del programma (verificare se il pagina proviene dal FORM stesso), l'altro per archiviare la chiave del record.
Di seguito la query

```
$sql = " SELECT * FROM rubrica WHERE id ='$_GET_SQL[id]' ";
$res = mysql_query($sql) or errore_db($sql);
```

```
if ($row = mysql_fetch_array($res))
{
      foreach($row AS $chiave => $valore) $row[$chiave] = htmlspecialchars ($valore);
}
```

Di seguito il FORM

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="yes">
<input type="hidden" name="id" value="<?php echo $_GET['id']; ?>">
<table width="500" cellspacing="0" cellpadding="0" border="0"    style="border: 1px solid #aaaaaa;">
<tr>
    <td width="250" valige="top">
    <table width="100%" cellspacing="5" cellpadding="2" border="0">
    <tr>
       <td>Nome</td>
       <td><input name="nome" value="<?php echo $row['nome']; ?>"></td>
    </tr>
    <tr>
       <td>Cognome</td>
       <td><input name="cognome" value="<?php echo $row['cognome']; ?>"></td>
   </tr>
    <tr>
       <td>Provincia</td>
       <td><select name="id_provincia">
       <option value="0"></option>
   <?php
   $query = "SELECT id, nome FROM provincie ORDER BY nome";
   $res = mysql_query($query) or errore_db($query);
   while ($row_p = mysql_fetch_array($res))
   {
     $nome = htmlspecialchars ($row_p['nome']);
     $id = $row_p['id'];
     $selected='';
     if ($id==$row['id_provincia']) $selected='selected';
     echo "\n<option $selected value=\"$id\">$nome</option>";
   }
   ?></select>
        </td>
    </tr>
    <tr>
       <td>Comune</td>
       <td><input name="comune" value="<?php echo $row['comune']; ?>"></td>
    </tr>
    <tr>
       <td>Indirizzo</td>
       <td><input name="indirizzo" value="<?php echo $row['indirizzo']; ?>"></td>
    </tr>
    </table>
    </td>
    <td width="250" valige="top">
    <table width="100%" cellspacing="5" cellpadding="2" border="0">
    <tr>
       <td>CAP</td>
```

```
        <td><input name="cap" value="<?php echo $row['cap']; ?>"></td>
    </tr>
     <tr>
        <td>Telefono</td>
        <td><input name="telefono" value="<?php echo $row['telefono']; ?>"></td>
    </tr>
     <tr>
        <td>Cellulare</td>
        <td><input name="cellulare" value="<?php echo $row['cellulare']; ?>"></td>
     </tr>
     <tr>
        <td>E-Mail</td>
        <td><input name="email" value="<?php echo $row['email']; ?>"></td>
     </tr>
     </table>
     </td>
</tr>
<tr>
     <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

Validazione del Form

Quando si inseriscono dei dati su un FORM, e buona cosa verificare i dati immessi. La verifica di un FORM lato client permette all'utente di inserire correttamente i dati, senza dover ricaricare la pagina. Vediamo alcuni semplici esempi di verifica:

- campo obbligatorio
- campo formattato (es. numero, data, e-mail, …)
- nome di persona o di oggetto (non contiene caratteri speciali)
- campo con vincoli nei valore (es. il valore deve essere compreso all'interno di un determinato intervallo di valori)
- campi con vincoli tra i loro valori (es. un valore di un campo non può essere maggiore del valore di un altro campo)

I valori formattati più comuni sono numero, data ed e-mail, che possono essere verificati mediante delle funzioni sviluppate appositamente. Vediamo di seguito il codice

```
function VerificaEmail(valore)
{
  if (valore == "") return true;
  var verifica =new RegExp("[^a-zA-Z0-9\\-\\.\\_\\@]");
  if (verifica.test(valore)) return false;
  verifica = new RegExp("^.+\\@(\\[?)[a-zA-Z0-9\\-\\.]+\\.([a-zA-Z]{2,3}|[0-9]{1,3})(\\]?)$");
  return verifica.test(valore);
}

function VerificaData(giorno, mese, anno)
{
  var data=new Date(anno, mese-1, giorno);
  if(data.getFullYear()==anno && data.getMonth()+1==mese && data.getDate()==giorno) return true;
  else return false;
```

```
}
```

Si noti che nella verifica dell'e-mail, abbiamo suddiviso il controllo in due fasi: verifica dei caratteri validi immessi e verifica della formattazione del testo.

Di seguito viene proposta una funzione molto utile

```
f function VerificaCaratteriAmmessi(valore, ammessi, minimo, massimo)
{
  if (valore == "") return true;
  var verifica =new RegExp("[^" + ammessi + "]");
  if (verifica.test(valore)) return false;
  verifica = new RegExp("([" + ammessi + "]{" + minimo + "," + massimo + "})");
  return verifica.test(valore);
}
```

Questa funzione riceve in ingresso il valore da controllare, i caratteri ammessi, il minimo ed il massimo numero di caratteri. Si noti che il valore può anche essere di lunghezza zero.

Continuando con le funzioni di verifica, vediamo come verificare i nomi di persone o oggetti

```
function VerificaNome(valore)
{
  return VerificaCaratteriAmmessi(valore, "a-zA-Z\\ ", 2, "");
}

function VerificaNomeNum(valore)
{
  return VerificaCaratteriAmmessi(valore, " a-zA-Z0-9\\ ", 2, "");
}
```

La prima funzione viene utilizzata per verificare l'immissione dei soli caratteri alfanumerici, mentre la seconda accetta anche caratteri numerici.

Per gestire anche i caratteri accentati (italiani), ecco altre due funzioni

```
function VerificaNomeIT(valore)
{
  return VerificaCaratteriAmmessi(valore, "a-zA-ZàòèéùìÀ\ ", 2, "");
}

function VerificaNomeNumIT(valore)
{
  return VerificaCaratteriAmmessi(valore, "a-zA-ZàòèéùìÀ0-9\\ ", 2, "");
}
```

Un caso particolare è la verifica di un indirizzo (che ammette anche la virgola per indicare il numero civico)

```
function VerificaIndirizzo(valore)
{
  return VerificaCaratteriAmmessi(valore, "a-zA-ZàòèéùìÀ0-9'.,\\ ", 2, "");
}
```

Anche il numero di telefono ha un suo insieme di caratteri

```
function VerificaTelefono(valore)
{
```

```
  return VerificaCaratteriAmmessi(valore, "\\\\0-9\\.,\\ \\-\\(\\)\\+", 3, "");
}
```

Finora abbiamo visto funzioni che verificano I valori, adesso vediamo la funzione generale che verifica il FORM inizialmente implementato

```
<script type="text/javascript">

function convalidaForm(form)
{
  var errori="";
  var primo_campo_errato;
  if (form.nome.value == "")
  {
    errori += "\n Il nome è obbligatorio";
    if (!primo_campo_errato) primo_campo_errato = form.nome;
  }
  if (!VerificaNomeIT(form.nome.value))
  {
    errori += "\n Il nome non è valido";
    if (!primo_campo_errato) primo_campo_errato = form.nome;
  }
  if (form.cognome.value == "")
  {
    errori += "\n Il cognome è obbligatorio";
    if (!primo_campo_errato) primo_campo_errato = form.cognome;
  }
  if (!VerificaNomeIT(form.cognome.value))
  {
    errori += "\n Il cognome non è valido";
    if (!primo_campo_errato) primo_campo_errato = form.cognome;
  }
  if (!VerificaNomeIT(form.comune.value))
  {
    errori += "\n Il comune non è valido";
    if (!primo_campo_errato) primo_campo_errato = form.comune;
  }
  if (!VerificaIndirizzo(form.indirizzo.value))
  {
    errori += "\n L'indirizzo non è valido";
    if (!primo_campo_errato) primo_campo_errato = form.indirizzo;
  }
  if (!VerificaEmail(form.email.value))
  {
    errori += "\n L'e-mail non è valida";
    if (!primo_campo_errato) primo_campo_errato = form.email;
  }
  if (errori != "")
  {
    alert("Attenzione:" + errori);
    primo_campo_errato.focus();
    return false;
  }
}
```

```
</script>
```

Si noti che in presenza di più campi non validi, viene visualizzata una finestra di avvertimento contenente l'elenco degli errori.

La funzione javascript ConvalidaForm() deve essere richiamata all'evento del SUBMIT del FORM.

```
<form onSubmit="javascript: return convalidaForm(this)" action="<?php echo $_SERVER['PHP_SELF']; ?>"
method="post">
```

Si noti la presenza di '*return'* nell'evento *OnSubmit*: la sua assenza, provoca l'esecuzione del FORM a prescindere dal valore di ritorno della funzione ConvalidaForm().
Si osservi che inoltre la funzione ConvalidaForm() non ha l'istruzione "return true;" perché in alcuni browser causa uno strano effetto doppio submit() (esegue due volte l'action del FORM).

Accessori

Questo capitolo mostra vari moduli accessori che un applicativo potrebbe avere, come caricare un documento tramite upload del file, creare pagine stampabili in HTML, esportare o importare in diversi formati quali CSV, Excel e XML, ed infine come inviare un e-mail. I moduli vengono detti accessori poiché non sono fini a se stessi ma rimangono comunque come parte necessaria di altri moduli più complessi.

Download e Upload di file

Download di file

Capita spesso di dover implementare un meccanismo di upload di file per inserire foto o documenti. Il primo esempio mostra come implementare un semplice form di caricamento dati.

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" enctype="multipart/form-data" method="post">
<input type="hidden" name="azione" value="si">
<table width="300" cellspacing="5" cellpadding="2" border="0" style="border: 1px solid #aaaaaa;">
<tr>
   <td>Allegato 1</td>
   <td><input type="file" name="file1"></td>
</tr>
<tr>
   <td>Allegato 2</td>
   <td><input type="file" name="file2"></td>
</tr>
<tr>
   <td>Allegato 3</td>
   <td><input type="file" name="file3"></td>
</tr>
<tr>
    <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

Nell'esempio viene implementato il caricamento di tre allegati: si noti che nel form è stato inserito "enctype="multipart/form-data"" necessario per l'upload dei file.

Nella seguente porzione di codice vengono mostrati in output i dati dell'upload. Si noti che tutte le informazioni relative ai file vengono archiviate nella variabile $_FILES.

```
if ($_POST['azione'])
{
 if (is_array($_FILES))
 {
  foreach ($_FILES as $nome_variabile => $vettore)
  {
     if ($vettore['error']) continue;
     echo "<br><br>campo = $nome_variabile";
     foreach ($vettore AS $chiave => $valore) echo "<br>$chiave = $valore";

  }
 }
}
```

Caricando tre file possiamo ottenere un risultato analogo al seguente:

```
campo = file1
name = esempio1.txt
type = application/octet-stream
tmp_name = C:\php\upload_tmp_dir\php1B.tmp
error = 0
```

```
size = 13985

campo = file2
name = esempio2.txt
type = audio/mpeg
tmp_name = C:\php\upload_tmp_dir\php\upload_tmp_dir\php1C.tmp
error = 0
size = 96384

campo = file3
name = esempio3.txt
type = audio/mpeg
tmp_name = C:\php\upload_tmp_dir\php\upload_tmp_dir\php1C.tmp
error = 0
size = 23118
```

In alcuni casi, (ad esempio per le foto) occorre lasciare inalterata l'estensione del file. Ovviamente prima di copiare il file $vettore['tmp_name'] nella cartella relativa ai documenti, occorre aggiungere un controllo sulle estensioni dei file: immaginatevi cosa potrebbe accadere se venisse caricato un file PHP.

Nell'esempio successivo vediamo come accettare solo file con le seguenti estensioni: jpg, gif, txt, doc, xml, pdf, doc, rtf, xls, png.

```
if ($_POST['azione'])
{
 if (is_array($_FILES))
 {
  $estensioni = array('jpg', 'gif', 'txt', 'doc', 'xml', 'pdf', 'doc', 'rtf', 'xls', 'png');
  foreach ($_FILES as $nome_variabile => $vettore)
  {
     if ($vettore['error']) continue;
     $suffisso = substr($vettore['name'], -3, 3);
     if (array_search($suffisso, $estensioni)!==false)
     {
         echo "<br>File $vettore[name]";
     }
  }
 }
}
```

È inevitabile che andando a posizionare tutti i file nella stessa cartella, ci possano essere più file con lo stesso nome: per evitare problemi occorre creare un meccanismo che crei dei file univoci con l'estensione corretta. Un metodo molto semplice è quello di utilizzare la data come nome del file. Per il calcolo della data, useremo la ben nota funzione per il calcolo dell'orario attuale.

```
function getmicrotime()
{
  list($usec, $sec) = explode(" ",microtime());
  return ((float)$usec + (float)$sec);
}
```

Se vengono inviati dei file molto piccoli, occorre aggiungere un contatore aggiuntivo, altrimenti avremo il rischio di avere lo stesso nome per alcuni di questi file. Vediamo nell'esempio.

```
  $i = 1;
  foreach ($_FILES as $nome_variabile => $vettore)
  {
```

```
    if ($vettore['error']) continue;
    $suffisso = substr($vettore['name'], -3, 3);
    if (array_search($suffisso, $estensioni)!==false)
    {
        echo "<br><br>File da spostare: $vettore[name]";
        $nome_file = (getmicrotime()*1000)."$i.$suffisso";
        echo "<br>Nuovo File: $nome_file";
    }
    $i++;
}
```

Ovviamente in tutti gli upload dei file si dovrebbe utilizzare lo stesso meccanismo.

L'ultima operazione è quella di spostare il file; in questo caso occorre utilizzare la funzione move_uploaded_file() che sposta il file assicurandosi che esso provenga effettivamente da un upload.

```
$i = 1;
foreach ($_FILES as $nome_variabile => $vettore)
{
    if ($vettore['error']) continue;
    $suffisso = substr($vettore['name'], -3, 3);
    if (array_search($suffisso, $estensioni)!==false)
    {
        $nome_file = (getmicrotime()*1000)."$i.$suffisso";
        if (move_uploaded_file($vettore['tmp_name'],percorso_principale."documenti/$nome_file"))
        {
            echo "<br>Upload completato: $nome_file";
        }
    }
    $i++;
}
```

N.B.: Si osservi che il calcolo del nome del file non garantisce che il file non sia già presente nella cartella.

Upload di file

Per ricevere un file dal programma ed aprirlo direttamente con l'applicativo corretto occorre manipolare l'intestazione del file.
Ecco un esempio per dare in download un file PDF, in modo che una volta scaricato venga immediatamente aperto dall'applicativo che legge i file PDF.

```
header('Content-Type: application/pdf');
header('Content-Length: '.$dimensione_file);
header('Content-disposition: inline; filename="'.$nome_file.'"');
……
// echo dei dati del file da inviare
```

La variabile $dimensione_file indica il numero la dimensione in byte del file, e $nome_file il nome del file. Nelle righe successive occorre dare in output i dati del file.
Nel successivo esempio vedremo come dare in output il file test.pdf.

```
header('Content-Type: application/pdf');
header('Content-Length: '.filesize('test.pdf'));
header('Content-disposition: inline; filename="test.pdf"');
readfile("test.pdf");
```

Nel successivo esempio vedremo come forzare il download del file 'test.pdf'.

```
if(isset($_SERVER['HTTP_USER_AGENT']) && strpos($_SERVER['HTTP_USER_AGENT'],'MSIE'))
        header('Content-Type: application/force-download');
else    header('Content-Type: application/octet-stream');
header('Content-Type: application/pdf');
header('Content-Length: '.filesize('test.pdf'));
header('Content-disposition: attachment; filename="test.pdf"');
readfile("test.pdf");
```

Esportazione per scambio dati

Nei paragrafi successivi verranno spiegati come implementare una libreria di oggetti utili per esportare ed importare dati in diversi formati quali CSV, XML e Excel. Per semplificare l'implementazione del codice dei primi due formati, verrà implementata una classe (che chiameremo FFile) che gestisce la lettura e scrittura dei file. Per una corretta gestione degli errori viene proposta una funzione simile a quella usata per le query.

```
function errore_generico($testo)
{
  global $parametri;
  if ($parametri['debug'] != 'no') echo "<br>Errore: $testo";
  if ($parametri['file_errori'])
  {
    $f = @fopen($parametri['file_errori'], 'a');
    @fwrite($f, "\n\n".date('d/m/Y')."\n Errore: $testo";
    @fclose($f);
  }
  exit();
}
```

In pratica la classe FFile si occupa principalmente di gestire tutte le verifiche di eventuali errori nell'uso delle funzioni di gestione del FileSystem, con l'obbiettivo di semplificare il codice del programma. Il trattamento dell'eventuale errore, viene relegato alla funzione errore_generico.
Di seguito il codice relativo alle variabili della classe

```
class FFile
{
    private $nome='';
    private $nome_completo='';
    private $f=false;
    private $stato=0;
     // 0 => nessun nome definito
     // 1 => nome definito e file inesistente
     // 2 => file chiuso
     // 3 => file aperto in lettura
     // 4 => file aperto in lettura/scrittura
     // 5 => file aperto in scrittura
……
}
```

La variabile $f memorizza la risorsa del file, e la variabile $stato tiene traccia delle informazioni relative al file. Tramite di esso avvengono tutti i controlli. Di seguito le funzioni membro relative ai controlli.

```
class FFile
{
……
    private function verifica_lettura($funzione)
    {
          if ( ($this->stato<3) || ($this->stato>4) ) errore_generico("$funzione: file non aperto in lettura");
    }

    private function verifica_scrittura($funzione)
    {
           if ($this->stato<4) errore_generico("$funzione: file non aperto in scrittura");
    }
```

```
    private function verifica_apertura($funzione)
    {
            if ($this->stato<3) errore_generico("$funzione: file non aperto");
    }

    private function verifica_nome($funzione)
    {
            if ($this->stato==0) errore_generico("$funzione: nome file non definito");
    }

    private function verifica_file($funzione)
    {
            if ($this->stato<2) errore_generico("$funzione: il file non esiste");
    }
……
}
```

Queste funzioni sono private, ed utilizzate in base all'occorrenza prima di ogni chiamata ad una funzione che opera nel FileSystem.

È possibile definire il nome del file al momento della creazione dell'oggetto.

```
    function __construct($file='')
    {
            $this->stato=0;
            if ($file)
            {
                if (is_file($file))
                {
                        $this->nome_completo = realpath($file);
                        $this->nome = basename($this->nome_completo);
                        $this->stato=2;
                }
                else
                {
                        $this->stato=1;
                        $this->nome_completo = $file;
                        $this->nome = $file;
                }
            }
     }
```

Alla chiusura dell'oggetto, se il file è ancora aperto, viene automaticamente chiuso.

```
    private function chiudi_se_aperto()
    {
            if ($this->stato>2)
            {
                 $this->fclose();
                 $this->stato=2;
            }
    }


     function __destruct()
     {
            $this->chiudi_se_aperto();
```

```
    }
```

Di seguito tutte le altre funzioni membro, che si occupano di gestire il file.

```
    public function fclose()
    {
        $this->verifica_apertura("fclose()");
        fclose($this->f);
        $this->stato=2;
    }

    public function feof()
    {
        $this->verifica_apertura("feof()");
        return feof($this->f);
    }

    public function fgetcsv($lunghezza='', $delimitatore=',', $chiusura='"')
    {
        $this->verifica_lettura("fgetc()");
        return fgetcsv($this->f, $lunghezza, $delimitatore, $chiusura);
    }

    public function fopen($mode='r')
    {
        $this->verifica_nome("fopen()");
        $this->chiudi_se_aperto();
        $this->f=@fopen($this->nome_completo, $mode);
        if (!$this->f) return false;
        if ($mode{0}=='r') $this->stato=3;
        else $this->stato=5;
        if ($mode{1}=='+') $this->stato=4;
        return true;
    }

    public function fputcsv($fields=array(), $delimiter=',',$chiusura='"')
    {
        $this->verifica_scrittura("fputcsv()");
        return fputcsv($this->f, $fields, $delimiter, $chiusura);
    }

    public function fread($length='')
    {
        $this->verifica_lettura("fread()");
        return fread($this->f, $length);
    }

    public function fwrite($testo)
    {
        $this->verifica_scrittura("fwrite()");
        return fwrite($this->f, $testo);
    }
}
```

Le funzioni membro implementate in questa classe, sono solo quelle che verranno effettivamente utilizzate negli esempi successivi. Come esercizio, il lettore può aggiungere tutte le altre funzioni che gestiscono il FileSystem.

formato CSV

Il formato CSV è un formato adatto per archiviare velocemente (e brutalmente) dei dati, poiché ha una struttura molto semplice e primitiva (molto in uso dai vecchi programmi). Solitamente viene utilizzato per importazioni ed esportazioni da un programma ad un altro o semplicemente per avere una copia di backup. In una esportazione, solitamente viene generato un file CSV per ogni tabella, dove ogni riga del file corrisponde a un record della tabella. all'interno di ogni riga, i valori sono divisi da un campo delimitatore (solitamente il carattere [;]), e possono anche essere racchiusi dalle virgolette. In questo ultimo caso, eventuali virgolette presenti nei valori sono preceduti dal carattere [\]. L'uso delle virgolette previene eventuali problemi dovuti alla possibile presenza del carattere delimitatore all'interno dei valori. La classe che andremo a definire (che chiameremo FCSV) implementa le funzionalità di scrittura e lettura su file in formato CSV. Questa classe dispone dei seguenti parametri opzionali:

- Delimitatore: definisce il carattere con il quale vengono separati i valori;
- Chiusura: definisce il carattere con il quale viene racchiuso il testo, qualora il testo contenesse il carattere delimitatore;
- Prima riga: definisce se la prima riga del file contiene i nomi dei campi;

L'implementazione di questa classe è relativamente semplice, l'unica difficoltà è rappresentata dall'opzione di scrivere la prima colonna nel file. L'utilità pratica di questa opzione e che nell'importazione dei dati l'oggetto è in grado di rilasciare un array associativo con le chiavi riprese dalla prima riga. Di seguito l'implementazione della classe.

```
class FCSV
{
    private $f=false;
    private $delimitatore=',';
    private $chiusura='"';
    private $struttura=array();
    private $prima_riga=false;

    public $r=array();
    public $nomi_prima_riga=false;

    function __construct($file='', $delimitatore=',', $chiusura='"')
    {
        $this->f=new FFile($file);
        $this->delimitatore=$delimitatore;
        $this->chiusura=$chiusura;
    }

    public function apri($mode='r')
    {
        $this->f->fopen($mode);
        $this->prima_riga=false;
        $this->struttura=array();
    }

    public function svuota()
    {
        foreach ($this->r as $chiave => $valore) $this->r[$chiave]='';
    }
```

```
    public function chiudi()
    {
         $this->f->fclose();
    }

    public function scrivi()
    {
        if ( ($this->nomi_prima_riga==true) && ($this->prima_riga==false) )
        {
            foreach ($this->r as $chiave => $valore) $this->struttura[]=$chiave;
            $this->prima_riga=true;
            $this->f->fputcsv($this->struttura, $this->delimitatore,$this->chiusura);
        }
        $this->f->fputcsv($this->r, $this->delimitatore,$this->chiusura);
        $this->svuota();
    }

    public function leggi()
    {
        if ($this->nomi_prima_riga==true)
        {
            if ($this->prima_riga==false)
            {
                $this->struttura=$this->f->fgetcsv(NULL, $this->delimitatore,$this->chiusura);
                $this->prima_riga=true;
            }
            $temp=$this->f->fgetcsv(NULL, $this->delimitatore,$this->chiusura);
            if (!$temp) return false;
            foreach ($temp as $chiave => $valore)
            {
                $this->r[$this->struttura[$chiave]]=$valore;
            }
            return true;
         }
        $this->r=$this->f->fgetcsv(NULL, $this->delimitatore,$this->chiusura);
        if ($this->r) return true;
        return false;
    }
}
```

Si noti che le informazioni delle opzioni di creazione del file CSV non sono memorizzate nello stesso. Vediamo con un esempio su come utilizzare questa classe. Nell'esempio seguente la lettura e scrittura di un file nel caso di non inserire i nomi dei campi nella prima riga.

```
<?php
// scrittura del file
$f=new FCSV('prova.txt');
$f->apri('w');
$f->r['campo1']="ciao1";
$f->r['campo2']="ciao2";
$f->r['campo3']="ciao3";
$f->scrivi();
$f->r['campo2']="salve";
$f->scrivi();
```

```
$f->chiudi();
?>
<br>
<table cellpadding="3" cellspacing="0" border="1">
<?php
// lettura del file
$f->apri('r');
$i=0;
while($f->leggi())
{
   if ($i==0)
   {
     echo "<tr>";
     foreach($f->r AS $chiave => $valore)
        echo "<td><b>$chiave</b></td>";
     echo "</tr>";
   }
   echo "<tr>";
   foreach($f->r AS $chiave => $valore)
      echo "<td>$valore</td>";
   echo "</tr>";
   $i++;
}
?>
</table>
```

Di seguito l'output ottenuto

0	**1**	**2**
ciao1	ciao2	ciao3
	salve	

Per inserire i nomi dei campi nella prima riga basta aggiungere la seguente istruzione

```
$f->nomi_prima_riga=true;
```

Di seguito l'output ottenuto

campo1	**campo2**	**campo3**
ciao1	ciao2	ciao3
	salve	

Si noti che nel primo esempio le chiavi sono numeriche, mentre nel secondo esempio vengono conservati i nomi dei campi.

formato XML

L'XML è in realtà un meta-linguaggio, da cui è possibile definire diversi linguaggi di formattazione. Questo formato è molto diffuso in Internet soprattutto per lo scambio dati. La classe che andremo a implementare utilizzata il linguaggio XML per definire un formato di esportazione dati, molto semplice. L'implementazione dell'esportazione dei dati è piuttosto semplice mentre l'importazione dei dati richiede una maggiore attenzione. Vediamo prima il codice relativo all'esportazione.

```
class FXML
```

```
{
   private $f=false;
   private $intestazione=false;
   private $mode;

   public $r=array();
   public $tabella='';

   function __construct($file='')
   {
        $this->f=new FFile($file);
   }

   public function apri($mode='r')
   {
        $this->f->fopen($mode);
        $this->intestazione=false;
        $this->mode=$mode;
   }

   public function svuota()
   {
        foreach ($this->r as $chiave => $valore) $this->r[$chiave]='';
   }

   public function chiudi()
   {
        if ( ($this->mode=='w') && ($this->intestazione==true) )
        {
            $this->f->fwrite("\n</dati>");
            $this->intestazione=false;
        }
        $this->f->fclose();
   }

   public function scrivi()
   {
        if ($this->intestazione==false)
        {
            $this->f->fwrite("<?xml version=\"1.0\" encoding=\"iso-8859-1\" ?".">\n<dati>");
            $this->intestazione=true;
        }
        $this->f->fwrite("\n\t<$this->tabella>");
        foreach ($this->r as $chiave => $valore)
        {
            $valore=str_replace("\n", " ", $valore);
            $valore=str_replace("\r", " ", $valore);
            $this->f->fwrite("\n\t\t<$chiave>$valore</$chiave>");
        }
        $this->f->fwrite("\n\t</$this->tabella>");
        $this->svuota();
   }
}
```

Come si può notare dal codice, l'esportazione di un file XML è piuttosto semplice da implementare. Per l'importazione dei dati verrà utilizzato il modulo PHP che offre il supporto expat. Questo modulo permette di creare parser XML e non richiede istallazioni aggiuntive.
Verrà implementato un parser XML che leggerà il file e archivierà i dati in una array temporaneo (FXML::dati). Per poter leggere file di grandi dimensioni è opportuno che il file lega una porzione per volta del file, e ad ogni lettura vengano utilizzati i dati letti. La funzione membro FXML::leggi() quindi utilizzerà i dati archiviati in FXML::dati per caricare l'array FXML::r utilizzato per visualizzare i dati. In pratica quando i sono stati letti tutti i dati archiviati, viene nuovamente richiamato il parser per continuare a leggere il file. Una ulteriore difficoltà è legata al problema dell'ultimo record di dati letto: infatti in file molto grandi la lettura di quest'ultimo da parte del parser, nella maggior parte dei casi non è mai tra la chiusura di un record e l'apertura del successivo, ma bensì all'interno di un record. Quindi ad ogni lettura, l'ultimo record letto è spesso incompleto e andrebbe inserito nell'array FXML::dati solo dopo che è stato archiviato completamente (alla successiva lettura del file).
Vediamo le funzioni e le variabili da aggiungere alla classe per implementare l'importazione.

```
class FXML
{
……
   private $livello=0;
   private $xml_parser;
   private $tabella_temp;
   private $campo_temp;
   private $elementi_temp;
   private $elementi_letti;
   private $dati_temp;
   private $dati;
……
   private function inizio_elemento($parser, $nome, $attributi)
   {
      $this->livello++;
      if ($this->livello==2)
      {
         $this->tabella_temp=$nome;
      }
      if ($this->livello==3)
      {
         $this->campo_temp=$nome;
         $this->dati_temp[$this->tabella_temp][$nome]='';
      }
   }

   private function fine_elemento($parser, $nome)
   {
      if ($this->livello==2)
      {
         $this->dati[$this->elementi_temp]=$this->dati_temp;
         unset($this->dati_temp);
         $this->elementi_temp++;
      }
      $this->livello--;
   }

   private function dati_elemento($parser, $data)
   {
      if ($this->livello==3)
```

```
    {
        $this->dati_temp[$this->tabella_temp][$this->campo_temp]=$data;
    }
}

public function leggi()
{
    if ($this->intestazione==false)
    {
        $this->livello=0;
        $this->xml_parser = xml_parser_create();
        xml_parser_set_option($this->xml_parser, XML_OPTION_CASE_FOLDING, true);
        xml_set_element_handler($this->xml_parser, array(&$this, 'inizio_elemento'), array(&$this,
            'fine_elemento') ) ;
        xml_set_character_data_handler($this->xml_parser, array(&$this, 'dati_elemento') );
        $this->elementi_temp=0;
        $this->elementi_letti=0;
        $this->intestazione=true;
    }

    if ($this->elementi_temp==$this->elementi_letti)
    {
        unset($this->dati);
        $this->elementi_temp=0;
        $this->elementi_letti=0;
        while ($this->elementi_temp==0)
        {
            if ($data = $this->f->fread(8192))
            {
                if (!xml_parse($this->xml_parser, $data, $this->f->feof()))
                {
                    errore_generico("Errore XML: ".
                        xml_error_string(xml_get_error_code($this->xml_parser)).
                        " alla linea %d".
                        xml_get_current_line_number($this->xml_parser), 'xml');
                }
            }
            else
            {
                xml_parser_free($this->xml_parser);
                return false;
            }
        }
    }
    $this->tabella=key($this->dati[$this->elementi_letti]);
    $this->r=&$this->dati[$this->elementi_letti][$this->tabella];
    $this->elementi_letti++;
    return true;
}
```

La creazione del parser è molto semplice, basta implementare la funzione di lettura del tag di apertura (FXML::inizio_elemento), la funzione di lettura del tag di chiusura (FXML::fine_elemento), ed infine la funzione di lettura del valore compreso tra l'apertura e la chiusura del tag (FXML::dati_elemento).

Attraverso la variabile FXML::livello siamo in grado di capire quale nodo stiamo esplorando, quindi di archiviare il nome della tabella al livello due, il nome del campo e il suo rispettivo valore a livello tre. La lettura del record viene temporaneamente memorizzata in FXML::dati_temp. Alla chiusura del tag che avviene a livello due, siamo in grado di capire che è stata completata la lettura del record FXML::dati_temp, che quindi può essere trasferita in FXML::dati. La variabile FXML::elementi_temp contiene il numero di record letti in modo completo e archiviati in FXML::dati. Si noti che nella funzione FXML::leggi(), c'è un ciclo che continua a leggere porzioni di file fino a quando non si è archiviato in modo completo almeno un record. Dopo aver letto un certo numero di record dal file, si cerca di utilizzarli con l'array FXML::r, utilizzando l'indice FXML::elementi_letti che tiene traccia dei record utilizzati.
Vediamo con un esempio su come utilizzare questa classe.

```
<?php
// scrittura del file
$f=new FXML('prova.xml');
$f->tabella="test";
$f->apri('w');
$f->r['campo1']="ciao1";
$f->r['campo2']="ciao2";
$f->r['campo3']="ciao3";
$f->scrivi();
$f->r['campo2']="salve";
$f->Scrivi();
$f->chiudi();
?>
<br>
<table cellpadding="3" cellspacing="0" border="1">
<?php
// lettura del file
$f->apri('r');
$i=0;
while($f->leggi())
{
   if ($i==0)
   {
     echo "<tr>";
     foreach($f->r AS $chiave => $valore) echo "<td><b>$chiave</b></td>";
     echo "</tr>";
   }
   echo "<tr>";
   foreach($f->r AS $chiave => $valore) echo "<td>$valore</td>";
   echo "</tr>";
   $i++;
}
?>
</table>
```

Di seguito l'output ottenuto

CAMPO1	CAMPO2	CAMPO3
ciao1	ciao2	ciao3
	salve	

Backup del Database

Con poche righe di codice è possibile eseguire il backup di un database in XML.

```
$f=new FXML('backup.xml');
$f->apri('w');

$ris = mysql_list_tables($conn_db['database'])
        or errore_generico("Impossibile elencare le tabelle");

while ($riga = mysql_fetch_row($ris))
{
   $f->tabella=$riga[0];
   echo "<br>Backup $f->tabella ";
   $sql = " SELECT * FROM `$f->tabella` ";
   $r_dati = mysql_query($sql) or errore_db($sql);
   while ($f->r = mysql_fetch_assoc($r_dati))
   {
      foreach ($f->r AS $chiave => $valore) $f->r[$chiave]=htmlspecialchars($valore);
      $f->scrivi();
   }
   mysql_free_result($r_dati);
}
mysql_free_result($ris);
$f->chiudi();
```

Scambio dati

Ipotizziamo di dover gestire una sincronizzazione di dati tra programmi. Ipotizziamo che il programma di esportazione dati, aggiorni un determinato file XML ed il programma client, si aggiorna prelevando ed elaborando questo file. Consideriamo anche il caso che la tabella da aggiornare contenga molti record, in questo caso, nella sincronizzazione occorre inviare solo le modifiche successive alla precedente esportazione.

Di seguito la struttura della tabella relativa all'esportazione.

```
CREATE TABLE `esportazione_dati` (
  `id` int(11) NOT NULL auto_increment,
  `ultima_modifica` datetime NOT NULL,
  `eliminato` char(1) NOT NULL,
  `campo1` varchar(200) NOT NULL,
  `campo2` varchar(200) NOT NULL,
  `campo3` varchar(200) NOT NULL,
  `campo4` varchar(200) NOT NULL,
  PRIMARY KEY  (`id`)
)
```

Nel campo "ultima_modifica" viene memorizzata la data dell'ultima modifica del record, e nel campo eliminato se uguale a 1 corrisponde ad un record eliminato. Ipotizzando che l'ultima esportazione sia avvenuta in data $ultima_esportazione, vediamo il codice relativo all'esportazione dei dati in XML.

```
   $f->tabella = "esportazione_dati";
   $sql = " SELECT * FROM `esportazione_dati` WHERE ultima_modifica >= '$ultima_esportazione' ";
   $r_dati = mysql_query($sql) or errore_db($sql);
   while ($f->r = mysql_fetch_assoc($r_dati))
   {
      foreach ($f->r AS $chiave => $valore) $f->r[$chiave]=htmlspecialchars($valore);
      $f->scrivi();
   }
```

```
mysql_free_result($r_dati);
$f->chiudi();
```

Per sicurezza è possibile inviare anche un valore di controllo, calcolato mediante un apposito algoritmo su tutti i dati del primo database. Alla fine della sincronizzazione si applica lo stesso algoritmo al secondo database e si confrontato i due risultati, se sono uguali, allora con molta probabilità i due database sono perfettamente allineati. Vediamo le modifiche alla procedura di esportazione.

```
$sql = " SELECT * FROM `esportazione_dati` WHERE eliminato = " ORDER BY id ";
$r_dati = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_assoc($r_dati))
{
   $testo = '';
   foreach ($row AS $chiave => $valore) $testo .= htmlspecialchars($valore);
   $controllo = md5($controllo.$testo);
}
mysql_free_result($r_dati);
$f->tabella = "esportazione_dati";
$sql = " SELECT * FROM esportazione_dati WHERE ultima_modifica >= '$ultima_esportazione' ";
$r_dati = mysql_query($sql) or errore_db($sql);
while ($f->r = mysql_fetch_assoc($r_dati))
{
   foreach ($f->r AS $chiave => $valore) $f->r[$chiave]=htmlspecialchars($valore);
   $f->scrivi();
}
mysql_free_result($r_dati);
$f->tabella = "controllo";
$f->r['controllo'] = $controllo;
$f->chiudi();
```

Adesso il codice relativo all'importazione

```
mysql_query(" START TRANSACTION ") or errore_db(" START TRANSACTION ");

$f->apri('r');
while($f->leggi())
{
   if ($f->tabella != 'esportazione_dati') break;
   $id = $f->r['id'];
   $campo1 = addslashes($f->r['campo1']);
   $campo2 = addslashes($f->r['campo2']);
   $campo3 = addslashes($f->r['campo3']);
   $campo4 = addslashes($f->r['campo4']);
   $campo5 = addslashes($f->r['campo5']);
   $eliminato = addslashes($f->r['eliminato']);
   if ($eliminato==1)
   {
      $sql = " DELETE FROM esportazione_dati WHERE id = '$id' ";
   }
   else
   {
      $sql = " SELECT id FROM esportazione_dati WHERE id = '$id' ";
      $res = mysql_query($sql) or errore_db($sql);
      if (mysql_num_rows($res)>0)
      {
```

```
        $sql = " UPDATE esportazione_dati
                SET campo1 = '$campo1',
                    campo2 = '$campo2',
                    campo3 = '$campo3',
                    campo4 = '$campo4',
                    campo5 = '$campo5'
                WHERE id = '$id' ";
      }
      else
      {
        $sql = " INSERT INTO esportazione_dati
                SET campo1 = '$campo1',
                    campo2 = '$campo2',
                    campo3 = '$campo3',
                    campo4 = '$campo4',
                    campo5 = '$campo5',
                    id = '$id' ";
      }
   }
   mysql_query($sql) or errore_db($sql);
}
$controllo = $f->r['controllo'];
$f->chiudi();

$sql = " SELECT * FROM esportazione_dati ORDER BY id ";
$r_dati = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_assoc($r_dati))
{
      $testo = '';
      foreach ($row AS $chiave => $valore) $testo .= htmlspecialchars($valore);
      $nuovo_controllo = md5($nuovo_controllo.$testo);
}
mysql_free_result($r_dati);

if ($nuovo_controllo == $controllo) mysql_query(" COMMIT ") or errore_db(" COMMIT ");
else
{
   echo "<p> Importazione dati fallita! </p> ";
   exit(0);
}
```

Si noti l'uso delle transazioni protette, per annullare le modifiche al database in caso di problemi.

Formato Excel

L'esportazione in Excel, avviene principalmente per dare la passibilità all'utente di avere la disponibilità dei dati all'interno di un foglio elettronico. In alcuni casi, questi file sono utilizzati per lo scambio dati tra utenti, come ad esempio inviare ad un cliente un listino prezzi.
La lettura e scrittura di file in questo formato, è diversa da quelli visti in precedenza, è molto più complessa, e per questo motivo è consigliato utilizzare delle librerie di terze parti che si possono trovare in Internet. Una delle più famose, è sicuramente Spreadsheet ReadExcel e Spreadsheet WriteExcel. La versione originale di WriteExcel richiede l'installazione di PEAR; tuttavia esiste una versione modificata di cui non necessità alcuna installazione aggiuntiva. Queste librerie possono essere ridistribuite o modificate sotto i termini di condizione della GNU Lesser General Public License. Questa licenza stabilisce il copyleft sul

singolo file di codice sorgente, ma non sull'intero software, cioè per esempio, del software rilasciato sotto licenza LGPL può essere incluso liberamente in un'applicazione licenziata sotto condizioni proprietarie, a patto che le modifiche apportate al codice sorgente del software stesso vengano rese pubbliche; tutti gli altri file dell'applicazione possono essere rilasciati con licenza proprietaria e senza codice sorgente.
Compatibilità con Microsoft Excel: in lettura versione Excel 97-2000 e 5.0/95; in scrittura versione 5.0/95.
Con queste librerie implementare una classe che esporta ed importa in Excel è piuttosto semplice.

```
class Fexcel
{
                private $file='';
                private $struttura=array();
                private $prima_riga=false;
                private $excel=null;
                private $temp_w=null;
                public $colonna_corrente=0;
                public $tabella='';

                public $r=array();

                function __construct($file='')
                {
                                $this->file=$file;
                }

                public function apri($mode='r')
                {
                   if ($mode=='w') $this->excel = new Spreadsheet_Excel_Writer($this->file);
                   else
    {
       $this->excel = new Spreadsheet_Excel_Reader();
       $this->excel->read($this->file);
       $this->excel->setOutputEncoding('CP1251');
    }
                                $this->prima_riga=false;
                                $this->colonna_corrente=0;
                                $this->struttura=array();
                }

                public function svuota()
                {
                                foreach ($this->r as $chiave => $valore) $this->r[$chiave]='';
                }

                public function chiudi()
                {
                                $this->excel->close();
                }

  public function scrivi()
  {
     if ($this->prima_riga==false)
     {
        $this->temp_w=$this->excel->addWorksheet($this->tabella);
        $i=0;
```

```
        foreach ($this->r as $chiave => $valore)
        {
           $this->struttura[]=$chiave;
           $this->temp_w->writeString($this->colonna_corrente, $i, $chiave);
           $i++;
        }
        $this->prima_riga=true;
        $this->colonna_corrente++;
     }
     $i=0;
     foreach ($this->r as $chiave => $valore)
     {
        $this->temp_w->writeString($this->colonna_corrente, $i, $valore);
        $i++;
     }
     $this->colonna_corrente++;
     $this->svuota();
  }

  public function leggi()
  {
     if ($this->colonna_corrente>$this->excel->sheets[0]['numRows']) return false;

     if ($this->prima_riga==false)
     {
        $i=1;
        $this->colonna_corrente=1;
        while($this->excel->sheets[0]['cells'][$this->colonna_corrente][$i])
        {
           $this->struttura[]=$this->excel->sheets[0]['cells'][$this->colonna_corrente][$i];
           $i++;
        }
        $this->colonna_corrente++;
        $this->prima_riga=true;
     }
     $i=1;
     foreach ($this->struttura as $chiave)
     {
       $valore=$this->excel->sheets[0]['cells'][$this->colonna_corrente][$i];
       $this->r[$chiave]=$valore;
       $i++;
     }
     $this->colonna_corrente++;
     return true;
  }
}
```

Vediamo un esempio di utilizzo.

```
<?php
// scrittura del file
$f=new FExcel('prova.xls');
$f->tabella="test";
$f->apri('w');
```

```
$f->r['campo1']="ciao1";
$f->r['campo2']="ciao2";
$f->r['campo3']="ciao3";
$f->scrivi();
$f->r['campo2']="salve";
$f->Scrivi();
$f->chiudi();
?>
<br>
<table cellpadding="3" cellspacing="0" border="1">
<?php
// lettura del file
$f->apri('r');
$i=0;
while($f->leggi())
{
   if ($i==0)
   {
     echo "<tr>";
     foreach($f->r AS $chiave => $valore)
        echo "<td><b>$chiave</b></td>";
     echo "</tr>";
   }
   echo "<tr>";
   foreach($f->r AS $chiave => $valore)
      echo "<td>$valore</td>";
   echo "</tr>";
   $i++;
}
?>
</table>
```

Esportazione dati per stampare

Oltre alla scambio dati, un utente potrebbe esportare dei dati anche per stamparli

Formato HTML

Il formato più semplice per effettuare una stampa è sicuramente l'HTML. Ovviamente per una corretta stampa di una pagina, l'utente dovrebbe configurare correttamente il browser: le impostazioni da tenere in considerazione sono:

- **Margini**: mettendo tutti i valori a zero vengono presi automaticamente i margini minimi della stampante;
- **Orientamento della pagina**: dipende dal tipo di stampa da fare, solitamente occorre controllarlo ogni volta che si stampa;
- **Intestazione e piè di pagina**: preferibilmente occorre mettere tutto a vuoto, altrimenti vengono stampati anche i riferimenti della pagina;
- **Stampa immagini e colori di sfondo:** devono essere selezionati altrimenti no vengono stampati.

Anche se questo formato è quello più semplice, spesso è quello meno preferito, in quanto le impostazioni appena elencate sono delegate all'utente e devono essere eseguite per ogni postazione dove si vuole stampare.
Per aprire automaticamente la finestra di stampa si può utilizzare la funzione javascript window.print()
Vediamo un semplice esempio di stampa.

```
Ciao a tutti!
<script type="text/javascript">
window.print();
</script>
```

Nell'esempio successivo vedremo come stampare due pagine aggiungendo una interruzione di pagina.

```
Prima pagina
<div style="page-break-before: always;"></div>
Seconda Pagina
<script type="text/javascript">
window.print();
</script>
```

Nel successivo esempio vedremo come risolvere il problema di stampare correttamente un tabulato utilizzando i fogli di stile. L'esempio riguarda la stampa dei contatti della rubrica (esempio visto nei capitoli precedenti).

```
<table style="border-collapse: collapse;" width="730" cellspacing="2" cellpadding="5" border="0">
<thead style="display: table-header-group;">
<tr bgcolor="#AAAAAA">
        <td width="120"><b>Cognome</b></td>
        <td width="120"><b>Nome</b></td>
        <td width="150"><b>Comune</b></td>
        <td width="160"><b>Indirizzo</b></td>
        <td width="180"><b>Recapiti Telefonici</b></td>
</tr>
</thead>
<tbody class="testo_small">
<?php
$colore1 = "#F0F0FF";
$colore2 = "#E0E0EF";
$colore_corrente = $colore1;
```

```
$sql = " SELECT r.*, p.nome AS provincia
           FROM rubrica AS r
           LEFT JOIN provincie AS p ON p.id = r.id_provincia
          ORDER BY r.cognome, r.nome ";
$res = mysql_query($sql) or errore_db($sql);
while ($row = mysql_fetch_array($res))
{
  $provincia = htmlspecialchars ($row['provincia']);
  $email = htmlspecialchars ($row['email']);
  $comune = htmlspecialchars ($row['cap']." ". $row['comune']);
  if ($provincia) $comune .= " ($provincia)";
?>
<tr bgcolor="<?php echo $colore_corrente; ?>">
  <td><?php echo htmlspecialchars($row['cognome']); ?></td>
  <td><?php echo htmlspecialchars($row['nome']); ?></td>
  <td><?php echo $comune; ?></td>
  <td><?php echo htmlspecialchars($row['indirizzo']); ?></td>
  <td><?php echo htmlspecialchars($row['telefono']." - ".$row['cellulare']); ?></td>
</tr>
<?php
  if ( $colore_corrente == $colore1 ) $colore_corrente = $colore2;
  else $colore_corrente = $colore1;
}
?>
</tbody>
</table>
<script type="text/javascript">
window.print();
</script>
```

Esempi di stame ed esportazioni

Nel capitolo precedente avevamo visto come implementare la gestione di una rubrica, adesso vediamo come si può aggiungere la stampa e l'esportazione.
Prima di vedere il sorgente occorre premettere che è possibile inserire il codice per la stampa direttamente all'interno della pagina relativa alla visualizzazione dei dati. Ovviamente prima di visualizzare il menu conviene calcolare i vari filtri ed utilizzarli anche per la query della stampa.
Vediamo il sorgente della prima parte della pagina

```
…

$massimo_righe=50;

$_POST_SQL = array_map('post_a_sql', $_POST);
$_POST = array_map('post_a_uso', $_POST);
$stampa = $_POST['stampa'];

If ($_POST['id_elimina'])
{
  $sql="DELETE FROM rubrica WHERE id = '$_POST_SQL[id_elimina]' ";
  mysql_query($sql) or errore_db($sql);
}

$sql_add = "";
```

```
if ($_POST_SQL['nome'])          $sql_add .= " AND r.nome LIKE '%$_POST_SQL[nome]%' ";
if ($_POST_SQL['cognome'])       $sql_add .= " AND r.cognome LIKE '%$_POST_SQL[cognome]%' ";
if ($_POST_SQL['email'])         $sql_add .= " AND r.email LIKE '%$_POST_SQL[email]%' ";
if ($_POST_SQL['comune'])        $sql_add .= " AND r.comune LIKE '%$_POST_SQL[comune]%' ";
if ($_POST_SQL['id_provincia']>0) $sql_add .= " AND r.id_provincia = '$_POST_SQL[id_provincia]' ";
if ($_POST_SQL['recapiti'])        $sql_add .= " AND ( (1=0) OR (r.telefono LIKE '%$_POST_SQL[recapiti]%')
OR (r.cellulare LIKE '%$_POST_SQL[recapiti]%') ) ";

$ordinamento=$_POST['ordinamento'];
if (!$ordinamento) $ordinamento='cognome_nome';

$modo=$_POST_SQL['modo'];
if (!$modo) $modo='ASC';

$order_by = '';
if ($ordinamento=="cognome_nome") $order_by = " ORDER BY r.cognome $modo, r.nome $modo ";
if ($ordinamento=="nome_cognome") $order_by = " ORDER BY r.nome $modo, r.cognome $modo ";
if ($ordinamento=="comune") $order_by = " ORDER BY r.comune $modo ";

if ($stampa=='si')
{
   include(percorso_principale.'include/top.php');

   ?>
   <center>
   <h3>Rubrica <?=date('d/m/Y')?></h3>
   <table style="border-collapse: collapse;" width="730" cellspacing="2" cellpadding="5" border="0">
   <thead style="display: table-header-group;">
   <tr bgcolor="#AAAAAA">
         <td width="120"><b>Cognome</b></td>
     <td width="120"><b>Nome</b></td>
         <td width="150"><b>Comune</b></td>
         <td width="160"><b>Indirizzo</b></td>
         <td width="180"><b>Recapiti Telefonici</b></td>
   </tr>
   </thead>
   <tbody class="testo_small">
   <?php
   $colore1 = "#F0F0FF";
   $colore2 = "#E0E0EF";
   $colore_corrente = $colore1;

   $sql = " SELECT r.*, p.nome AS provincia
         FROM rubrica AS r
         LEFT JOIN provincie AS p ON p.id = r.id_provincia
         WHERE (1=1) $sql_add
         $order_by ";
   $res = mysql_query($sql) or errore_db($sql);
   while ($row = mysql_fetch_array($res))
   {
     $provincia    = htmlspecialchars ($row['provincia']);
     $email                  = htmlspecialchars ($row['email']);
         $comune     = htmlspecialchars ($row['cap']." ". $row['comune']);
```

```
    if ($provincia) $comune .= " ($provincia)";
  ?>
  <tr bgcolor="<?php echo $colore_corrente; ?>">
   <td><?php echo htmlspecialchars($row['cognome']); ?></td>
   <td><?php echo htmlspecialchars($row['nome']); ?></td>
   <td><?php echo $comune; ?></td>
   <td><?php echo htmlspecialchars($row['indirizzo']); ?></td>
   <td><?php echo htmlspecialchars($row['telefono']." - ".$row['cellulare']); ?></td>
  </tr>
  <?php
        if ( $colore_corrente == $colore1 ) $colore_corrente = $colore2;
   else $colore_corrente = $colore1;
  }
  ?>
  </tbody>
  </table>
  <script type="text/javascript">
  window.print();
  </script>
  </center>
  <?php
  include(percorso_principale.'include/bottom.php');
  exit();
}

include(percorso_principale.'include/top_rubrica2.php');
....
```

Si noti che per riconoscere l'esecuzione della stampa abbiamo utilizzato il campo $_POST['stampa']. Quindi nel form occorre aggiungere un campo nascosto:

```
<input type="hidden" name="stampa" value="">
```

Per richiamare la stampa abbiamo bisogno di una funzione javascript

```
function StampaHtml()
{
 var d = document.filtri;
 d.target = '_blank';
 d.stampa.value = 'si';
 d.submit();
 d.target = '';
 d.stampa.value = '';
}
```

Si noti che la proprietà target e il campo stampa vengono ripristinati in modo da non creare problemi con il successivo filtro.

Infine vediamo il sorgente per aggiungere il bottone di stampa

```
$menu = new menu_orizzontale();
$menu->bottone('lista_contatti', 'Lista<br>Contatti', 'index.php', '', 'Visualizza la lista dei contatti');
$menu->bottone('nuovo_contatto', 'Nuovo<br>Contatto', "javascript: ModificaContatto('')", '',
                    'Crea un nuovo contatto');
$menu->bottone('stampa', 'Stampa<br>Contatti', 'javascript: StampaHtml()', '', 'Stampa la lista dei contatti');
echo $menu->scrivi_html();
```

Inviare una E-Mail

Per inviare una E-Mail solitamente viene utilizzata la funzione mail(), poiché di default il PHP utilizza sendmail (normalmente presente in tutti i sistemi Linux/Unix) per utilizzare questa funzione anche nei sistemi Windows occorre configurare correttamente il php.ini. I parametri da impostare sono rispettivamente:

- **SMTP:** nome DNS o indirizzo IP del server SMTP;
- **smtp_port:** numero della porta del server specificato da SMTP (il valore predefinito è 25) ;
- **sendmail_from:** vostra e-mail.

Solitamente nei sistemi Linux/Unix durante la compilazione, il percorso di sendmail viene trovato e impostato. Qualora ciò non dovesse accadere, è possibile specificare il percorso nel parametro sendmail_path del php.ini.

Di seguito un semplice esempio sull'uso di questa funzione

```
mail("pippo@esempio.com", "Oggetto", "Questo è un testo\r\nLinea 2\r\nLinea 3");
```

Nell'esempio successivo vedremo come inviare una e-mail con intestazioni supplementari:

```
mail($email_destinatario, $oggetto, $messaggio,
    "From: $mittente <$email_mittente>\r\n" .
    "Reply-To: $mittente <$email_mittente>\r\n" .
    "X-Mailer: $nome_programma");
```

Ecco un altro esempio con più destinatari e messaggio in HTML

```
$destinatari  = "$nome_dest1 <$email_dest1>, $nome_dest2 <$email_dest2>";
$oggetto = "Questo è un oggetto";

$messaggio = '<html>
<body>
<p><b>Questo è il testo del messaggio in HTML</b></p>
</body>
</html>';

/* Per inviare email in formato HTML, si deve impostare l'intestazione Content-type. */
$intestazioni  = "MIME-Version: 1.0\r\n";
$intestazioni .= "Content-type: text/html; charset=iso-8859-1\r\n";

$intestazioni .= "To: $nome_dest3 <$email_dest3>, $nome_dest4 <$email_dest4>\r\n";
$intestazioni .= "From: $mittente <$email_mittente>\r\n";

mail($destinatari, $oggetto, $messaggio, $intestazioni);
```

Vediamo come modificare l'intestazione per aggiungere la conferma di lettura.

```
$intestazioni  = "MIME-Version: 1.0\r\n";
$intestazioni .= "Content-type: text/html; charset=iso-8859-1\r\n";
$intestazioni .= "To: $nome_dest3 <$email_dest3>, $nome_dest4 <$email_dest4>\r\n";
$intestazioni .= "From: $mittente <$email_mittente>\r\n";
$intestazioni .= "Disposition-Notification-To: $email_mittente\r\n";
```

Semplice form di invio Mail

L'esempio seguente riguarda la creazione di un semplice form per inviare una E-Mail. In pratica tutti i parametri necessari per inviare l'e-mail vengono inseriti sul form.
Di seguito il codice del form.

```
<form  onSubmit="javascript: return convalidaForm(this)" action="<?php echo $_SERVER['PHP_SELF']; ?>"
        method="post">
<input type="hidden" name="azione" value="si">
<table width="450" cellspacing="5" cellpadding="2" border="0" style="border: 1px solid #aaaaaa;">
<tr>
   <td>Mittente</td>
   <td><input  size="40" name="mittente"></td>
</tr>
<tr>
   <td>E-Mail mittente</td>
   <td><input  size="40" name="email_mittente"></td>
</tr>
<tr>
   <td>E-Mail destinatario</td>
   <td><input  size="40" name="email_destinatario"></td>
</tr>
<tr>
   <td>Oggetto</td>
   <td><input  size="40" name="oggetto"></td>
</tr>
<tr>
     <td align="center" colspan="2">
       <div align="left">Testo</div>
       <textarea style="width:95%;height:100px" name="messaggio_html"></textarea>
     </td>
</tr>
<tr>
     <td align="center" colspan="2"><input value="  Invia  " type="submit"></td>
</tr>
</table>
</form>
<script type="text/javascript">
function convalidaForm(form)
{
  var errori="";
  var primo_campo_errato;
  if (form.email_mittente.value == "")
  {
    errori += "\n L'E-Mail del mittente è obbligatorio";
    if (!primo_campo_errato) primo_campo_errato = form.email_mittente;
  }
  if (!VerificaEmail(form.email_mittente.value))
  {
    errori += "\n L'E-Mail del mittente non è valida";
    if (!primo_campo_errato) primo_campo_errato = form.email_mittente;
  }
  if (form.email_destinatario.value == "")
```

```
  {
    errori += "\n L'E-Mail del destinatario è obbligatorio";
    if (!primo_campo_errato) primo_campo_errato = form.email_destinatario;
  }
  if (!VerificaEmail(form.email_destinatario.value))
  {
    errori += "\n L'E-Mail del destinatario non è valida";
    if (!primo_campo_errato) primo_campo_errato = form.email_destinatario;
  }
  if (form.oggetto.value == "")
  {
    errori += "\n L'oggetto è obbligatorio";
    if (!primo_campo_errato) primo_campo_errato = form.oggetto;
  }
  if (errori != "")
  {
    alert("Attenzione:" + errori);
    primo_campo_errato.focus();
    return false;
  }
}
</script>
```

Infine il codice relativo all'invio della e-mail.

```
if ($_POST['azione'])
{
   $nome_programma = 'Mailer Fantastico';
   $_POST = array_map('post_a_uso', $_POST);

   $intestazioni .= "From: $_POST[mittente] <$_POST[email_mittente]>\r\n".
     "Reply-To: $_POST[mittente] <$_POST[email_mittente]>\r\n".
     "X-Mailer: $nome_programma\r\n".
     "MIME-version: 1.0\r\n".
     "Content-type: multipart/mixed;\r\n".
     "\tboundary=\"Message-Boundary\"\r\n\r\n";

   $messaggio = "--Message-Boundary\r\n".
     "Content-type: text/html; charset=\"iso-8859-1\"\r\n".
     "Content-transfer-encoding: 8BIT\r\n\r\n";
   $messaggio .= $_POST['messaggio_html'];
   $messaggio .= "\r\n\r\n--Message-Boundary\r\n";

   mail($_POST['email_destinatario'], $_POST['oggetto'], $messaggio, $intestazioni);
}
```

E-mail con allegato

Per inserire un allegato occorre codificare l'allegato in una stringa MIME base64 in quanto il corpo di una e-mail non supporta dati binari a 8 bit.

```
$dimensione_file = @filesize($nome_file);
$f = @fopen($nome_file, "r");
$contenuto = @fread($f, $dimensione_file);
@fclose($f);
$dati_codificati = chunk_split(base64_encode($contenuto));
```

Un'altra informazione che ci occorre per allegare un file è il suo tipo MIME, che può essere calcolata in base all'estensione. Vediamo la funzione che lo calcola

```
Function tipo_mime($nome_file = '')
{
   $file_partizionato = split("\.", basename($nome_file));
   $estensione = $file_partizionato[1];
   $mime = array(
    'hqx'   =>  'application/mac-binhex40',
    'cpt'   =>  'application/mac-compactpro',
    'doc'   =>  'application/msword',
    'bin'   =>  'application/macbinary',
    'oda'   =>  'application/oda',
    'pdf'   =>  'application/pdf',
    'ai'    =>  'application/postscript',
    'eps'   =>  'application/postscript',
    'ps'    =>  'application/postscript',
    'smi'   =>  'application/smil',
    'smil'  =>  'application/smil',
    'mif'   =>  'application/vnd.mif',
    'xls'   =>  'application/vnd.ms-excel',
    'ppt'   =>  'application/vnd.ms-powerpoint',
    'wbxml' =>  'application/vnd.wap.wbxml',
    'wmlc'  =>  'application/vnd.wap.wmlc',
    'dcr'   =>  'application/x-director',
    'dir'   =>  'application/x-director',
    'dxr'   =>  'application/x-director',
    'dvi'   =>  'application/x-dvi',
    'gtar'  =>  'application/x-gtar',
    'php'   =>  'application/x-httpd-php',
    'php4'  =>  'application/x-httpd-php',
    'php3'  =>  'application/x-httpd-php',
    'phtml' =>  'application/x-httpd-php',
    'phps'  =>  'application/x-httpd-php-source',
    'js'    =>  'application/x-javascript',
    'swf'   =>  'application/x-shockwave-flash',
    'sit'   =>  'application/x-stuffit',
    'tar'   =>  'application/x-tar',
    'tgz'   =>  'application/x-tar',
    'xhtml' =>  'application/xhtml+xml',
    'xht'   =>  'application/xhtml+xml',
    'zip'   =>  'application/zip',
    'mid'   =>  'audio/midi',
    'midi'  =>  'audio/midi',
```

```
    'mpga'  =>  'audio/mpeg',
    'mp2'   =>  'audio/mpeg',
    'mp3'   =>  'audio/mpeg',
    'aif'   =>  'audio/x-aiff',
    'aiff'  =>  'audio/x-aiff',
    'aifc'  =>  'audio/x-aiff',
    'ram'   =>  'audio/x-pn-realaudio',
    'rm'    =>  'audio/x-pn-realaudio',
    'rpm'   =>  'audio/x-pn-realaudio-plugin',
    'ra'    =>  'audio/x-realaudio',
    'rv'    =>  'video/vnd.rn-realvideo',
    'wav'   =>  'audio/x-wav',
    'bmp'   =>  'image/bmp',
    'gif'   =>  'image/gif',
    'jpeg'  =>  'image/jpeg',
    'jpg'   =>  'image/jpeg',
    'jpe'   =>  'image/jpeg',
    'png'   =>  'image/png',
    'tiff'  =>  'image/tiff',
    'tif'   =>  'image/tiff',
    'css'   =>  'text/css',
    'html'  =>  'text/html',
    'htm'   =>  'text/html',
    'shtml' =>  'text/html',
    'txt'   =>  'text/plain',
    'text'  =>  'text/plain',
    'log'   =>  'text/plain',
    'rtx'   =>  'text/richtext',
    'rtf'   =>  'text/rtf',
    'xml'   =>  'text/xml',
    'xsl'   =>  'text/xml',
    'mpeg'  =>  'video/mpeg',
    'mpg'   =>  'video/mpeg',
    'mpe'   =>  'video/mpeg',
    'qt'    =>  'video/quicktime',
    'mov'   =>  'video/quicktime',
    'avi'   =>  'video/x-msvideo',
    'movie' =>  'video/x-sgi-movie',
    'doc'   =>  'application/msword',
    'word'  =>  'application/msword',
    'xl'    =>  'application/excel',
    'eml'   =>  'message/rfc822'
  );
  $risultato = $mime[strtolower($estensione)];
  return ( ! isset($risultato) ? 'application/octet-stream' : $risultato);
}
```

Si osservi che se l'estensione non rientra nell'elenco viene restituito 'application/otet-stream'.

Per semplificare l'inserimento degli allegati possiamo creare una funzione

```
function InserisciAllegato($percorso_file, $nome_file, $disposition='attachment')
{
 $dimensione_file = @filesize($percorso_file);
 $f = @fopen($percorso_file, "r");
```

```
  $contenuto = @fread($f, $dimensione_file);
  @fclose($f);
  $dati_codificati = chunk_split(base64_encode($contenuto));
  $tipo_file = tipo_mime($nome_file);
  return "Content-type: $tipo_file; name=\"$nome_file\"\r\n".
     "Content-Transfer-Encoding: base64\r\n".
     "Content-disposition: $disposition; filename=\"$nome_file\"\r\n\r\n".
     "$dati_codificati\r\n\r\n";
}
```

I parametri di questa funzione sono rispettivamente:

- **$percorso_file:** il percorso completo del file da allegare;
- **$nome_file:** il nome del file che deve comparire come allegato;
- **$disposition:** solitamente se il file è un allegato si inserisce "attachment", ma nel caso delle immagini che devono essere visualizzate in linea si imposta "inline".

Ecco un esempio di invio e-mail con allegato

```
$intestazioni .= "From: $mittente <$email_mittente>\r\n" .
    "Reply-To: $mittente <$email_mittente>\r\n" .
    "X-Mailer: $nome_programma\r\n" .
    "MIME-version: 1.0\r\n" .
    "Content-type: multipart/mixed\r\n" .
    "\tboundary=\"Message-Boundary\"\r\n ";


$messaggio = "--Message-Boundary\r\n".
  "Content-type: text/html; charset=\"iso-8859-1\"\r\n".
  "Content-transfer-encoding: 8BIT\r\n";
$messaggio .= $messaggio_html;
$messaggio .= "--Message-Boundary\r\n";
$messaggio .= InserisciAllegato("foto.jpg", "foto.jpg", 'inline');
$messaggio .= "--Message-Boundary\r\n";
$messaggio .= InserisciAllegato("documento.doc", "documento.doc");

mail($email_destinatario, $oggetto, $messaggio, $intestazioni);
```

Sorgente esempio per invio Mail con allegato

L'esempio seguente riguarda la creazione di un semplice form per inviare una E-Mail con allegato. Rispetto al form precedente c'è in più il campo relativo al file da inviare ed aggiungere l'attributo enctype al tag FROM.

Di seguito le modifiche al codice del form visto in precedenza.

```
<form  onSubmit="javascript: return convalidaForm(this)" action="<?php echo $_SERVER['PHP_SELF']; ?>"
        enctype="multipart/form-data" method="post">
……
……
<tr>
   <td>Allegato (*)</td>
   <td><input type="file" size="40" name="file_da_allegare"></td>
</tr>
……
……
</form>
```

```
<p>(*) Allegati ammessi:  'jpg', 'gif', 'txt', 'doc', 'xml', 'pdf', 'doc', 'rtf', 'xls', 'png'</p>
……
……
<script type="text/javascript">
function convalidaForm(form)
{
……
……
  if (form.file_da_allegare.value == "")
  {
    errori += "\n L'allegato è obbligatorio";
    if (!primo_campo_errato) primo_campo_errato = form.file_da_allegare;
  }
……
……
```

Al codice relativo all'invio della e-mail occorre anche gestire l'allegato. Di seguito il codice

```
if ($_POST['azione'])
{
 $percorso_file = '';
 $nome_file = '';
 $disposition = 'attachment';
 if (is_array($_FILES))
 {
  $estensioni = array('jpg', 'gif', 'txt', 'doc', 'xml', 'pdf', 'doc', 'rtf', 'xls', 'png');
  $immagini = array('jpg', 'gif', 'png');
  foreach ($_FILES as $nome_variabile => $vettore)
  {
    if ($vettore['error']) continue;
    $suffisso = substr($vettore['name'], -3, 3);
    if (array_search($suffisso, $estensioni)!==false)
    {
      $percorso_file = $vettore['tmp_name'];
      $nome_file = $vettore['name'];
      if (array_search($suffisso, $immagini)!==false) $disposition = 'inline';
    }

  }
 }
 if ($nome_file)
 {
  $nome_programma = 'Mailer Fantastico';
  $_POST = array_map('post_a_uso', $_POST);

  $intestazioni .= "From: $_POST[mittente] <$_POST[email_mittente]>\r\n".
   "Reply-To: $_POST[mittente] <$_POST[email_mittente]>\r\n".
   "X-Mailer: $nome_programma\r\n".
   "MIME-version: 1.0\r\n".
   "Content-type: multipart/mixed;\r\n".
   "\tboundary=\"Message-Boundary\"\r\n\r\n";

  $messaggio = "--Message-Boundary\r\n".
   "Content-type: text/html; charset=\"iso-8859-1\"\r\n".
```

```
      "Content-transfer-encoding: 8BIT\r\n\r\n";
    $messaggio .= $_POST['messaggio_html'];
    $messaggio .= "\r\n\r\n--Message-Boundary\r\n";
    $messaggio .= InserisciAllegato($percorso_file, $nome_file, $disposition);

    mail($_POST['email_destinatario'], $_POST['oggetto'], $messaggio, $intestazioni);
  }
  else
  {
     echo "<p><b>formato di file non valido!</b></p>";
  }
}
```

PHPMailer

Per spedire una e-mail mediante un server remoto SMTP/IMAP occorre instaurare una connessione mediante socket al server. Senza entrare nei dettagli in questo tipo di programmazione, è possibile utilizzare direttamente una delle tante librerie Open Source che si trovano sulla rete; una di queste è PHPMailer. Questa libreria permette di inviare una email tramite la funzione mail oppure mediate un server remoto SMTP/IMAP con relativa autenticazione o connessione protetta o mediante sendamail.
L'esempio successivo mostra come utilizzare la classe PHPMailer.

```
include_once('class.phpmailer.php');

$mail = new PHPMailer();
$messaggio = '<html>
<body>
<p><b>Questo è il testo del messaggio in HTML</b></p>
</body>
</html>';

// tipo di connessione
$mail->IsMail();
// fine tipo di connessione

$mail->From      = "nome@tuodominio.com";
$mail->FromName   = "Tuo nome";
$mail->Subject    = "Test soggetto PHPMailer via smtp";
$mail->AltBody    = "Per visualizzare correttamente il messaggio devi utilizzare un visualizzatore HTML!";
$mail->MsgHTML($messaggio);
$mail->AddAddress("destinatario@altrodominio.com", "Nome destinatario");
$mail->AddAttachment("img/phpmailer.gif");
if(!$mail->Send()) echo "Errore Mailer: " . $mail->ErrorInfo;
else  echo "Messaggio spedito correttamente!";
```

L'esempio successivo mostra come utilizzare la classe PHPMailer per inviare la stessa e-mail mediante server SMTP.

```
……
……
// tipo di connessione
$mail->IsSMTP();
$mail->Host      = "mail.tuodominio.com";
$mail->SMTPAuth   = true;
```

```
$mail->Username = "tuausername";
$mail->Password = "tuapassword";
// fine tipo di connessione
……
……
```

Si noti che nell'esempio avviene un collegamento al server con autenticazione; se il server accetta connessioni anonime allora le seguenti istruzioni possono essere omesse:

```
$mail->SMTPAuth   = true;
$mail->Username = "tuausername";
$mail->Password = "tuapassword";
```

In caso si debba utilizzare una porta diversa da quella di default (porta 25) è possibile impostarla mediante la seguente istruzione:

```
$mail->Port = 465;
```

In caso di connessione protetta SSL basta aggiungere l'istruzione seguente:

```
$mail->SMTPSecure = "ssl";
```

Altro parametro importante è la conferma di lettura:

```
$mail->ConfirmReadingTo = "nome@tuodominio.com";
```

Sorgente esempio per invio Mail con allegato con PHPMailer

Di seguito le eventuali modifiche da apportare al sorgente visto nell'esempio precedente.

```
if ($_POST['azione'])
{
  $percorso_file = '';
  if (is_array($_FILES))
  {
    $estensioni = array('jpg', 'gif', 'txt', 'doc', 'xml', 'pdf', 'doc', 'rtf', 'xls', 'png');
    foreach ($_FILES as $nome_variabile => $vettore)
    {
       if ($vettore['error']) continue;
       $suffisso = substr($vettore['name'], -3, 3);
       if (array_search($suffisso, $estensioni)!==false)
       {
         $percorso_file = $vettore['tmp_name'];
       }

    }
  }
  if ($percorso_file)
  {

    $_POST = array_map('post_a_uso', $_POST);

    $mail = new PHPMailer();

    // tipo di connessione
    $mail->IsMail();
    // fine tipo di connessione
```

```
    $mail->From       = $_POST['email_mittente'];
    $mail->FromName   = $_POST['mittente'];
    $mail->Subject    = $_POST['oggetto'];
    $mail->AltBody    = "Per visualizzare correttamente il messaggio devi utilizzare un visualizzatore HTML!";
    $mail->MsgHTML($_POST['messaggio_html']);
    $mail->AddAddress($_POST['email_destinatario']);
    $mail->AddAttachment($percorso_file);
    if(!$mail->Send()) echo "Problemi con la spedizione della E-Mail: " . $mail->ErrorInfo;
    else  echo "Messaggio spedito correttamente!";
  }
  else
  {
     echo "<p><b>formato di file non valido!</b></p>";
  }
}
```

Si noti che nell'esempio abbiamo utilizzato sendamail, per esercizio potete modificare il sorgente per inviare la stessa e-mail tramite server SMTP.

Autenticazione

Questo capitolo illustra l'implementazione del modulo di autenticazione. Come vedremo nel corso dei vari esempi, l'autenticazione non solo svolge il compito di creare degli accessi riservati agli utenti registrati, ma anche di riconoscere l'utente, e caricare tutti i sui parametri di configurazione del programma.
Anche se la maggior parte delle soluzioni proposte per migliorare la sicurezza e l'usabilità del modulo di autenticazione sono concepiti per i gestionali, tuttavia la base ed alcune soluzioni possono essere utilizzate per l'autenticazione di portali web.

Autenticazione HTTP

Questo tipo di autenticazione consiste nell'utilizzare il web server APACHE per chiedere le credenziali all'utente. Ci sono due metodi, il primo è attraverso la creazione dei file ".htaccess" e ".htpasswd", che vanno inseriti nella directory da proteggere. Il secondo è attraverso il PHP mediante l'uso della funzione header() e necessita l'installazione del PHP come modulo e non come CGI.
Queste tecniche sono molto semplice da implementare, e sono utilizzate in alcuni casi specifici:

- Nei gestionali, per la creazione di più livelli di autenticazione;
- Nei siti web, per proteggere l'area di amministrazione, senza dover necessariamente implementare sofisticati moduli di autenticazione.

Prenderemo in esame solo la prima tecnica poiché se lo si vuole aggiungere ad un applicativo, non occorre modificare alcuna pagina PHP.
L'esempio seguente mostra un esempio su come creare i due file ".htaccess" e ".htpasswd".
File .htaccess

```
AuthUserFile "/percorso/.htpasswd"
AuthGroupFile /dev/null
AuthName "Accesso Riservato"
AuthType Basic

<Limit GET>
require user utente1
require user utente2
require user utente3
…..
</Limit>
```

"AuthUserFile" indica il percorso completo del file .htpasswd; "AuthName" indica il titolo della finestra; "AuthType" indica il tipo di autenticazione e "require user" indica il nome utente che può accedere.
Mediante il comando htpasswd è possibile creare il file .htpasswd. Vediamo il codice per creare il file:

```
touch .htpasswd
htpasswd -mb .htpasswd utente1 password1
htpasswd -mb .htpasswd utente2 password2
htpasswd -mb .htpasswd utente3 password3
…..
```

Semplice modulo di autenticazione

Nell'esempio seguente si cercherà di implementare un semplice modulo di autenticazione.
Poiché l'autenticazione è il processo in cui il programma identifica l'utente, è o necessario disporre di una tabella dove memorizzare gli utenti. Di seguito la struttura di tale tabella.

```
CREATE TABLE `utenti` (
  `id` int(11) NOT NULL auto_increment,
  `username` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `email` varchar(255) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `username` (`username`)
)
```

L'autenticazione inizia quando il programma chiede i dati di accesso. L'inserimento di questi dati avviene tramite un semplice FORM. Di seguito un esempio.
File index.php

```
<?php
include('include/parametri.php');
include(percorso_principale.'include/top.php');
?>
<center>
<form action="main.php" method="post">
<table cellspacing="2" cellpadding="2" border="0">
<tr>
  <td>Username</td>
  <td><input size="20" name="Username"></td>
</tr>
<tr>
  <td class="testo_normal">Password</td>
  <td><input size="20" name="Password" type="password" value=""></td>
</tr>
  <tr>
  <td colspan="2" align="center"><input type="submit" value="ENTRA"></td>
</tr>
</table>
</form>
</center>
<?php
include(percorso_principale.'include/bottom.php');
?>
```

É di assoluta importanza impostare il metodo del FORM a **POST**, altrimenti si vedrebbe la password nella barra degli indirizzi.
Si noti come il campo input della password è di tipo "password", per impedire la lettura della stessa.
Il file funzioni.php e connessione.php sono identici a quelli utilizzati nei precedenti capitoli.

Come vedremo in seguito, le informazioni relative al riconoscimento dell'utente, vengono memorizzate nelle variabili di sessione. Quest'ultima viene creata al momento del riconoscimento dell'utente.
Di seguito due semplici file dimostrativi dell'area protetta, l'accesso a questa area è consentito solo attraverso l'autenticazione.

main.php

```
<?php
include('include/parametri.php');
include(percorso_principale.'include/connessione.php');
include(percorso_principale.'include/funzioni.php');
include(percorso_principale.'include/autenticazione.php');
include(percorso_principale.'include/top.php');
?>
Pagina 1
| <a href="main2.php">Vai alla Pagina 2</a>
| <a href="index.php?logout=yes">Esci</a>
<?php
include(percorso_principale.'include/bottom.php');
?>
```

main2.php

```
<?php
include('include/parametri.php');
include(percorso_principale.'include/connessione.php');
include(percorso_principale.'include/funzioni.php');
include(percorso_principale.'include/autenticazione.php');
include(percorso_principale.'include/top.php');
?>
Pagina 2
| <a href="main.php">Vai alla Pagina 1</a>
| <a href="index.php?logout=yes">Esci</a>
<?php
include(percorso_principale.'include/bottom.php');
?>
```

I due file mostrano la semplicità della navigazione all'interno dell'area protetta.
Si noti che per introdurre una pagina all'interno dell'area protetta, è necessario aggiungere l'istruzione di inclusione del file autenticazione.php.
Per uscire dall'are protetta occorre distruggere la sessione in corso, questa operazione può essere eseguita nel file 'index.php'. Di seguito le modifiche al file index.php

```
<?php
include('include/parametri.php');

if ($_GET['logout'])
{
   session_start();
   session_unset();
   session_destroy();
}

include(percorso_principale.'include/top.php');
?>
……..
```

Tutta la procedura di controllo dell'area riservata avviene nel file autenticazione.php. In pratica ha un doppio utilizzo, se richiamato dal FORM della pagina index.php, crea una sessione per memorizzare le variabili di autenticazione, altrimenti se inclusa da una pagina dell'area protetta, verifica che la corretta autenticazione dell'utente. Di seguito il sorgente.

```
<?php
session_start();
```

```
if (!$_SESSION['id_user'])
{
    $Username = post_a_sql($_POST['Username']);
    $Password = post_a_sql($_POST['Password']);

    if ((!$Username) || (!$Password))
   {
            header("Location: index.php");
            exit(0);
   }

   $query = " SELECT *
                FROM utenti
                WHERE username = '$Username'
               AND password = '$Password' ";
   $risultato = mysql_query($query) or errore_db($query);
   if (mysql_num_rows($risultato)==0)
   {
            header("Location: index.php");
            exit(0);
   }
  $row = mysql_fetch_assoc($risultato);
  $_SESSION['id_user'] = $row['id'];
  mysql_free_result($risultato);
}
?>
```

Gestione delle password

Password criptate

Ci sono tre ottime ragioni per hashare una password.

1. in caso di attacco con SQL injection o con altri mezzi si ottiene un lista di codici hash;
2. in ogni server ci sono dei gestori o admin che sono degli umani e quindi potenzialmente corruttibili o raggirabili mediante l'ingegneria sociale;
3. la privacy obbliga per legge a custodire in modo sicuro i dati riservati (backup compresi);

Riscriviamo quindi il file autenticazione.php

```
<?php
session_start();

if (!$_SESSION['id_user'])
{
    $Username = post_a_sql($_POST['Username']);
    $Password = post_a_sql($_POST['Password']);
    if ((!$Username) || (!$Password))
   {
          header("Location: index.php");
          exit(0);
   }

   $query = " SELECT *
                FROM utenti
                WHERE username = '$Username'
               AND password = MD5('$Password') ";
   $risultato = mysql_query($query) or errore_db($query);
   if (mysql_num_rows($risultato)==0)
   {
          header("Location: index.php");
          exit(0);
   }
  $row = mysql_fetch_assoc($risultato);
  $_SESSION['id_user'] = $row['id'];
  mysql_free_result($risultato);
}?>
```

Password dimenticate

Immaginiamoci che l'utente si scordi la password del suo account. Una tecnica molto semplice nei gestionali aziendali, è quella di inviargli per e-mail una nuova password (non è possibile risalire alla password preesistente in quanto sono memorizzate in codice hash). Occorre quindi avere nel database degli utenti anche l'e-mail (come campo obbligatorio). Il motivo per cui questa tecnica è applicabile in un gestionale ma non in un portale, è che come vedremo in seguito la password dell'utente viene cambiata al momento della richiesta.

Di seguito la nuova struttura dati degli utenti.

```
CREATE TABLE `utenti` (
`id` INT NOT NULL AUTO_INCREMENT,
`username` VARCHAR( 50 ) NOT NULL ,
`password` VARCHAR( 50 ) NOT NULL,
```

```
`email` VARCHAR( 250 ) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `username` (`username`)
);
```

L'utente inserisce l'username in un FORM apposito.

```
……..
<form action="proponi_password.php" method="post">
<table cellspacing="2" cellpadding="2" border="0">
<tr>
      <td>Username</td>
      <td><input size="20" name="Username"></td>
</tr>
  <tr>
      <td colspan="2" align="center"><input type="submit" value="Ricorda Password"></td>
</tr>
</table>
</form>
……
```

Poiché il calcolo della nuova password viene generata in automatico, occorre implementare una funzione che calcola una stringa di caratteri validi, con un minimo numero di lettere ($min). Questa funzione può essere implementata all'interno del file funzioni.php

```
function testo_casuale($min)
{
      $i = 0;
      $risultato = "";
      while($i<$min)
     {
            $carattere = chr(rand(48,122));
            if (      ( ($carattere >= 'a') && ($carattere <= 'z') )
                  || ( ($carattere >= 'A') && ($carattere <= 'Z') )
                  || ( ($carattere >= '0') && ($carattere <= '9') )   )
            {
                  $risultato .= $carattere;
                  $i++;
            }
     }
     return $risultato;
}
```

Nella pagina 'proponi_password.php', il sistema preleva l'e-mail dell'utente, quindi determina la nuova password, ed una volta spedita, aggiorna l'anagrafica degli utenti.

```
……..
<?php
$errore = "";
$parametri['oggetto_email'] = "Il tuo gestionale preferito";
$parametri['messaggio_email'] = "Ciao <username>, ti abbiamo generato una nuova password: <password>.
Una volta entrato nel portale, potrai cambiarla con un'altra di tuo gradimento.";
$Username = post_a_sql($_POST['Username']);
$query = " SELECT * FROM utenti WHERE username = '$Username' ";
$risultato = mysql_query($query) or errore_db($query);
if ($row = mysql_fetch_assoc($risultato))
{
       $email = $row['email'];
```

```
        if ($email)
        {
                $nuova_password = testo_casuale(10);
                $messaggio = str_replace('<username>', $Username, $parametri['messaggio_email']);
                $messaggio = str_replace('<password>', $nuova_password, $messaggio);
                if (@mail($email, $parametri['oggetto_email'], $messaggio))
                {
                        $query = " UPDATE utenti
                                        SET password = '$nuova_password'
                                        WHERE username = '$Username' ";
                        mysql_query($query) or errore_db($query);
                }
                else
                {
                         $errore = "Non sono riuscito a spedire l'e-mail";
                }
        }
        else
        {
           $errore = "Utente <b>$Username</b> non ha un email valida ";
        }
}
else
{
   $errore = "Username <b>$Username</b> non presente in anagrafica";
}

?>
…….
```

Solitamente i vari parametri del portale come $parametri['oggetto_email'] e $parametri['messaggio_email'], possono essere gestiti nel file di configurazione parametri.php.

Come aumentare la sicurezza

Al nostro esempio visto in precedenza, è possibile aggiungere diverse impostazioni di sicurezza

- restrizioni di accesso sui giorni della settimana e sull'orario
- restrizioni di accesso su determinati indirizzi IP
- monitoraggio degli accessi degli utenti
- monitoraggio dei tentativi falliti di autenticazione
- restrizioni di accesso su un numero massimo di tentativi falliti in un determinato lasso di tempo

Per aggiungere le restrizioni di accesso del primo punto, occorre aggiungere alcuni campi alla tabella utenti. Di seguito la struttura della tabella

```
CREATE TABLE `utenti` (
`id` INT NOT NULL AUTO_INCREMENT,
`username` VARCHAR( 250 ) NOT NULL ,
`password` VARCHAR( 250 ) NOT NULL,
`email` VARCHAR( 250 ) NOT NULL ,
`accesso_ristretto` CHAR( 1 ) DEFAULT '1',
`accesso_settimana` SET('Domenica', 'Lunedì', 'Martedì', 'Mercoledì', 'Giovedì', 'Venerdì', 'Sabato') NOT NULL
DEFAULT 'Domenica,Lunedì,Martedì,Mercoledì,Giovedì,Venerdì,Sabato',
`accesso_am_dalle` TIME DEFAULT '07:00:00',
`accesso_am_alle` TIME DEFAULT '15:00:00',
`accesso_pm_dalle` TIME DEFAULT '15:00:00',
`accesso_pm_alle` TIME DEFAULT '20:00:00',
PRIMARY KEY (`id`),
UNIQUE KEY `username_2` (`username`),
KEY `username` (`username`)
);
```

Con il campo accesso_ristretto, indichiamo con valore '1', se l'utente deve essere sottoposto a restrizione, altrimenti valore vuoto. Con il campo acesso_settimana indichiamo in quali giorni della settimana l'utente può accedere. Infine con gli altri campi di tipo TIME, indichiamo in quali intervalli di tempo l'utente può accedere durante la giornata.
Utilizzando un vettore $lista_settimana contenente i giorni della settimana, e con date('w'), possiamo determinare il giorno attuale con $lista_settimana[date('w')]. La lista dei giorni in cui è possibile accedere è memorizzata nella stringa del campo accesso_settimana della tabella utenti. Con la funzione esplode() possiamo copiare quest'ultima in un vettore $lista. Se il giorno attuale ($lista_settimana[date('w')]) è all'interno di quest'ultimo, allora l'accesso è consentito.
Vediamo di seguito le modifiche da apportare al file

```
<?php
session_start();

if (!$_SESSION['id_user'])
{
    $Username = post_a_sql($_POST['Username']);
    $Password = post_a_sql($_POST['Password']);
    if ((!$Username) || (!$Password))
   {
            header("Location: index.php");
            exit(0);
   }

   $query = " SELECT *
```

```
                FROM utenti
                WHERE username = '$Username'
               AND password = MD5('$Password') ";
  $risultato = mysql_query($query) or errore_db($query);
  if (mysql_num_rows($risultato)==0)
  {
        header("Location: index.php");
        exit(0);
  }
  $row = mysql_fetch_assoc($risultato);
  if ($row['accesso_ristretto']==1)
  {
    $verifica_giorno = true;
    $lista = explode(',', $row['accesso_settimana']);
    $lista_settimana = array('Domenica', 'Lunedì', 'Martedì', 'Mercoledì', 'Giovedì', 'Venerdì', 'Sabato');
    if (!in_array($lista_settimana[date('w')], $lista)) $verifica_giorno = false;

    $ora_attuale=date('H:i:00');
    $verifica_ora = false;
    if ( ($ora_attuale >=$row['accesso_am_dalle']) && ($ora_attuale <=$row['accesso_am_alle']) )
       $verifica_ora = true;
    if ( ($ora_attuale >=$row['accesso_pm_dalle']) && ($ora_attuale <=$row['accesso_pm_alle']) )
       $verifica_ora = true;

     if ( ($verifica_giorno == false) || ($verifica_ora == false) )
     {
        header("Location: index.php");
        exit(0);
     }
  }
  $_SESSION['id_user'] = $row['id'];
  mysql_free_result($risultato);
}
?>
```

Per le restrizioni su determinati indirizzi IP, ci sono due opzioni: se il numero di indirizzi è al massimo quattro o cinque, basterebbe inserire i rispettivi campi nella tabella utenti, altrimenti occorre creare una nuova tabella. Consideriamo solo la seconda ipotesi, e vediamo come implementare la tabella restrizioni_indirizzi_ip

```
CREATE TABLE `restrizioni_indirizzi_ip` (
  `id` int(11) NOT NULL auto_increment,
  `id_utente` int(11) NOT NULL,
  `indirizzo` varchar(30) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `id_utente` (`id_utente`)
)
```

La determinazione dell'indirizzo IP dell'utente tramite getenv("REMOTE_ADDR"), conviene inserirla prima della query di controllo username e password. Vedremo in seguito in quali altri modi utilizzare la variabile $indirizzo_ip.

```
$indirizzo_ip = getenv("REMOTE_ADDR");
```

Di seguito il codice della restrizione.

```
$query = " SELECT sum(1) AS Num
            FROM restrizioni_indirizzi_ip
            WHERE id_utente = '$row[id]'
            AND '$indirizzo_ip' LIKE indirizzo ";
$r_IP = mysql_query($query) or errore_db($query);
if (mysql_num_rows($r_IP)==0)
{
      header("Location: index.php");
      exit(0);
}
$row_IP = mysql_fetch_assoc($r_IP);
if ( (!$row_IP['Num']) || ($row_IP['Num']==0) )
{
      header("Location: index.php");
      exit(0);
}
mysql_free_result($r_IP);
```

Vediamo un elenco di esempi dei valori del campo indirizzo della tabella restrizioni_indirizzi_ip

%	L'utente può entrare da qualsiasi indirizzo
192.168.1.%	L'utente può entrare da indirizzi che iniziano per 192.168.1.
188.21.134.67	L'utente può entrare dall'indirizzo 188.21.134.67

È ovvio che se la tabella restrizioni_indirizzi_ip, non contiene indirizzi per l'utente, allora quest'ultimo non può entrare.

Per monitorare gli accessi degli utenti, occorre avere una tabella dove memorizzare il riferimento dell'utente, l'indirizzo IP e la data.

```
CREATE TABLE `accessi_utente` (
   `id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
   `id_utente` INT NOT NULL ,
   `indirizzo` VARCHAR( 30 ) NOT NULL ,
   `data` DATETIME NOT NULL ,
   KEY `id_utente` (`id_utente`)
)
```

Questi dati vengono archiviati dopo l'autenticazione. Vediamo di seguito il codice

```
$query = " INSERT INTO accessi_utente ( id_utente, indirizzo, data )
            VALUES ( '$_SESSION[id_user]', '$indirizzo_ip', Now() ) ";
mysql_query($query) or errore_db($query);
```

Per monitorare i tentativi falliti, occorre avere una tabella dove memorizzare il riferimento dell'utente, l'indirizzo IP, la data e il codice del motivo del fallimento.

```
CREATE TABLE `accessi_negati_utente` (
  `id` INT(11) NOT NULL auto_increment,
  `username` VARCHAR( 250 ) NOT NULL ,
  `indirizzo` VARCHAR(30) NOT NULL,
  `data` DATETIME NOT NULL,
  `motivo` INT(11) NOT NULL,
  PRIMARY KEY  (`id`),
KEY `username` (`username`)
)
```

Vediamo come implementare una semplice funzione che archivia il tentativo fallito.

```
function accesso_negato($indirizzo_ip, $Username, $motivo)
```

```
{
    $query = " INSERT INTO accessi_negati_utente (username, indirizzo, data, motivo )
                VALUES ( '$Username', '$indirizzo_ip', Now(),'$motivo' ) ";
    mysql_query($query) or errore_db($query);
}
```

La funzione viene chiamata in ogni punto di negazione di accesso, prima della seguente istruzione

```
header("Location: index.php");
```

Per limitare gli attacchi a forza bruta, e possibile applicare una restrizione sul numero massimo di tentativi falliti (definito nella variabile $parametri['max_tentativi']), all'interno di un determinato lasso di tempo (definito nella variabile $parametri['tempo_tentativi']). È consigliabile definire queste variabili in un file di configurazione parametri.php, o definirli in una tabella nel database; la definizione delle impostazioni del programma e quelle personalizzate, verranno spiegato successivamente. La restrizione deve essere inserita prima della query di controllo username e password. Vediamo di seguito il codice.

```
$query = " SELECT sum(1) AS Num
            FROM accessi_negati_utente
            WHERE username = '$Username'
            AND ADDDATE(data, INTERVAL $parametri[tempo_tentativi] MINUTE)>=Now() ";
$risultato = mysql_query($query) or errore_db($query);
if (mysql_num_rows($risultato)>0)
{
    $row = mysql_fetch_assoc($risultato);
    if ($row['Num']>$parametri['max_tentativi'])
    {
        accesso_negato($indirizzo_ip, $Username, 4);
        header("Location: index.php");
        exit(0);
    }
}
mysql_free_result($risultato);
```

Con tutte queste restrizioni, è consigliabile dare qualche indicazione all'utente, cioè inviare una variabile $errore contenente il riferimento del problema alla pagina index.php. Quest'ultima si occuperà di gestire l'informazione all'utente. Vediamo il codice per inviare la variabile alla pagina.

```
header("Location: index.php?errore=1");   // username o password errati
```

Impostazioni del programma

Le impostazioni del programma sono tutti quei parametri utilizzati dal programma che vengono utilizzati a prescindere dall'utente connesso, al contrario le impostazioni dell'utente sono quei parametri che dipendono dall'utente connesso. Questa differenza, determina una differente gestione a database. Vediamo la struttura delle due tabelle

```
CREATE TABLE `parametri_sistema` (
    `nome` VARCHAR( 20 ) NOT NULL ,
    `valore` VARCHAR( 255 ) NOT NULL ,
    PRIMARY KEY ( `nome` )
);

CREATE TABLE `parametri_utente` (
  `id` int(11) NOT NULL auto_increment,
  `id_utente` int(11) NOT NULL,
  `nome` varchar(20) NOT NULL,
  `valore` varchar(255) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `id_utente` (`id_utente`)
);
```

Poiché tutte le impostazioni personalizzate dell'utente sono precedute dalla verifica dell'autenticazione, è normale inserire la sua implementazione nel file autenticazione.php. Per completezza, nello stesso file possono essere inserite anche le impostazioni del programma. Si osservi che tutte queste impostazioni si intendono utilizzate all'interno dell'area protetta; se occorre avere delle impostazioni all'esterno di questa area è possibile utilizzare il file parametri.php.
Di seguito le righe di codice da aggiungere nel file autenticazione.php.

```
$query = " SELECT * FROM parametri_sistema ";
$risultato = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_assoc($risultato))
{
     $parametri[$row['nome']] = $row['valore'];
}
mysql_free_result($risultato);

if (!$_SESSION['id_user'])
{
     ……
     ……
}

$query = " SELECT *
            FROM parametri_utente
            WHERE id_utente = '$_SESSION[id_user]' ";
$risultato = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_assoc($risultato))
{
     $parametri_utente[$row['nome']] = $row['valore'];
}
mysql_free_result($risultato);
```

Usabilità

Ricorda password

Ci sono tanti metodi per migliorare l'usabilità della fase di autenticazione. Il loro uso non è sempre indicato in quanto spesso vanno in contrasto con la sicurezza.
Tra questi vi è sicuramente, quando il programma dà la possibilità di ricordare user e password memorizzando tali dati nei cookies.
L'utente deve poter decidere se accettare questa facilitazione, in quanto vengono memorizzate sulla postazione dei dati sensibili.
Un metodo classico consiste nell'inserire due checkbox per decidere se ricordare username e password nel FORM di autenticazione: in base a cosa di seleziona vengono memorizzati i dati nei cookies.
La prima modifica da fare file index.php consiste nel cercare nei cookies, eventuali username e password memorizzate

```
$user_ricorda = post_a_uso($_COOKIE['user_ricorda']);
$pwd_ricorda = post_a_uso($_COOKIE['pwd_ricorda']);
```

quindi il FORM viene riscritto in questo modo

```
<form action="main.php" method="post">
<table cellspacing="2" cellpadding="2" border="0">
<tr>
      <td>Username</td>
      <td><input value="<?=$user_ricorda?>" size="20" name="Username"></td>
</tr>
<tr>
      <td class="testo_normal">Password</td>
      <td><input value="<?=$pwd_ricorda?>" size="20" name="Password" type="password" value=""></td>
</tr>
<tr>
      <td class="testo_normal">Ricorda</td>
      <td>[<input type="checkbox" <?php if ($user_ricorda) echo "checked"; ?> name="ricorda_user"
            value="1"> User]  [<input type="checkbox" <?php if ($pwd_ricorda) echo "checked"; ?>
            name="ricorda_pwd" value="1"> Pwd]
</tr>
<tr>
      <td colspan="2" align="center"><input type="submit" value="ENTRA"></td>
</tr>
</table>
</form>
```

Si noti come gli checkbox rimangano selezionati se alla precedente autenticazione l'utente li aveva fleggati.

La memorizzazione dei dati nei cookies avviene nel file autenticazione.php alla file di tutti i controlli di accesso. Di seguito il codice

```
if ($ricorda_user==1)
{
        setcookie ("user_ricorda", post_a_uso($_POST['Username']), time() + 604800);
}
else
{
        setcookie ("user_ricorda", "", time() - 3600);
}
```

```
if ($ricorda_pwd==1)
{
        setcookie ("pwd_ricorda", post_a_uso($_POST['Password']), time() + 604800);
}
else
{
  setcookie ("pwd_ricorda", "", time() - 3600);
}
```

Si noti che il periodo in cui viene impostato ogni cookie è di una settimana (60x60x24x7=604800).

Selezionare l'utente

In caso di gestionali con un numeri limitato di utenti, è possibile selezionare l'utente da una <SELECT>.

```
<tr>
 <td>Username</td>
 <td><select name="Username">
   <option value=""></option>
   <?php
   $query = " SELECT username FROM utenti ORDER BY username ";
   $res = mysql_query($query) or errore_db($query);
   while ($row = mysql_fetch_array($res))
   {
       $nome = htmlspecialchars ($row['username']);
       $selected='';
       if ($user_ricorda==$nome) $selected='selected';
       echo "\n<option $selected value=\"$nome\">$nome</option>";
   }
   ?>
   </select>
 </td>
</tr>
```

Poiché l'utente è in grado di visualizzare l'elenco completo degli utenti, è ovvio che pone un altro problema di sicurezza. È consigliabile utilizzare questo meccanismo solo su gestionali che lavorano in aree riservate. Una variante grafica alla <SELECT> è l'uso di una griglia a scorrimento con immagine e nome dell'utente, e si lascia l'implementazione al lettore, come esercizio.

Programmazione avanzata

Questo capitolo illustra alcuni aspetti più complessi della programmazione. Verranno presi in esame alcuni aspetti implementati relativi all'archiviazione di dati statici (immagini, testi o query), all'implementazione dei FORM creati dinamicamente e diverse soluzioni in javascript per avere funzionalità particolari.

Archiviazione di dati statici

In applicativo molto complesso potrebbe essere necessario disporre di archivi statici di immagini, testi o query. A prescindere da quali siano questi dati, è importante determinare un meccanismo per recuperare i dati che servono nella maniera più veloce possibile. Ovviamente per dati statici si intende dati in cui le eventuali modifiche (che potrebbero avvenire al runtime) possono essere utilizzate al momento ma non memorizzate.
Mentre la gestione dei parametri che necessita l'archiviazione delle eventuali modifiche avviene tramite database, la gestione di archivi statici avviene tramite un semplice meccanismo di inclusione di file php.
Una tecnica molto semplice è quella di organizzare i dati in tanti file PHP. Ovviamente occorre distribuire i dati nei var file, in modo tale che in esecuzione ne debbano caricarsi pochi. In oltre occorre un meccanismo semplice per caricare automaticamente i file. Infine è utile che le chiavi di ricerca siano "parlanti" in modo da rendere leggibile il sorgente che utilizza tali dati.

Archivio multi lingua

L'esempio più classico di archivio statico è sicuramente l'archivio per gestire più lingue. Nei sorgenti seguenti viene presentato un meccanismo molto semplice di archiviazione che può essere utilizzato anche per qualunque altro tipo di archivio.
Il meccanismo si basa sul memorizzare i dati mediante degli array. Ogni dato ha una chiave composta da un prefisso identico al nome del file ed un codice alfanumerico che identifica il dato all'interno del file. È ovvio che il codice alfanumerico deve ricordare il dato associato. In oltre, in ogni file deve essere definita un variabile per determinare se il file è stato già caricato.
Vediamo come esempio il file 'ana.php' che archivia dei dati

```
$archivi_caricati['ana']=1;

$diz['ana_indirizzo']= 'Indirizzo';
$diz['ana_rag_sociale']= 'Ragione sociale';
$diz['ana_recapito']= 'Recapito';
$diz['ana_partita_iva']= 'Partita IVA';
$diz['ana_codice_fiscale']= 'Codice Fiscale';
$diz['ana_titolo']= 'Titolo';
```

A questo punto è facile implementare un meccanismo per caricare il file PHP in base al codice.

```
function linguaggio($identificativo)
{
   global $diz, $identificati_caricati, $len;
   $id = substr($identificativo, 0, 3);
   if (!$archivi_caricati[$id])
   {
      if (!is_file(percorso_principale."linguaggi/$len/$id.php"))
            errore_generico("Dizionario non trovato: $identificativo");
      include(percorso_principale." linguaggi/$len/$id.php");
   }
   return $diz[$identificativo];
}
```

Si osservi che in questo caso la variabile $len identifica la cartella relativa alla lingua.

Nell'esempio l'istruzione per stampare l'etichetta 'Ragione sociale'

```
echo linguaggio('ana_rag_sociale');
```

Dall'esempio appena visto è evidente il vantaggio di questa tecnica rispetto all'archiviazione su database:

- Il caricamento di un file PHP è più veloce di un esecuzione di una query;
- La visibilità di dati con molto testo è maggiore su un file PHP rispetto alla visualizzazione di un programma di gestione database (questi dati possono essere gestiti da database in modo agevole solo se si implementano dei specifici pannelli di controllo);
- Nel database la ricerca è più veloce se la chiave è un numero, ma questo comporterebbe una pessima leggibilità del sorgente.

Form dinamici

In questo paragrafo vedremo come implementare un FORM con campi dinamici.
Il primo esempio, riguarda la semplice creazione dinamica di 10 campi input.

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="si">
<table cellspacing="2" cellpadding="5" border="0">
<?php
for ($i=0; $i<10; $i++)
{
?>
<tr>
  <td><b>Campo <?php echo $i; ?>:</b></td>
  <td><input name="Valore[<?php echo $i; ?>]"></td>
</tr>
<?php
}
?>
<tr>
     <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

I nomi dei campi vengono chiamati come elementi di un vettore; questo semplifica sia la creazione del form e sia l'estrazione dei valori. Vediamo come è semplice estrarre i valori.

```
<table cellspacing="2" cellpadding="5" border="0">
<?php
   $Valore = &$_POST['Valore'];
   for ($i=0; $i<10; $i++)
   {
?>
<tr>
  <td><b>Campo <?php echo $i; ?>:</b></td>
  <td><?php echo $Valore[$i]; ?></td>
</tr>
<?php
}
?>
</table>
```

Nel prossimo esempio vedremo come implementare il FORM a partire dai campi memorizzati in una tabella. Di seguito la struttura della tabella.

```
CREATE TABLE `campi_form` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
`etichetta` VARCHAR( 50 ) NOT NULL
)
```

La tabella contiene l'etichetta che verrà visualizzata nel FORM. La soluzione presentata di seguito imposta le chiavi dei campi come indice del vettore.

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="si">
<table cellspacing="2" cellpadding="5" border="0">
<?php
```

```
 $query = "SELECT * FROM campi_form ORDER BY etichetta";
 $res = mysql_query($query) or errore_db($query);
 while ($row = mysql_fetch_array($res))
 {
    $etichetta = htmlspecialchars ($row['etichetta']);
    $id = $row['id'];
?>
<tr>
 <td><b><?php echo $etichetta; ?>:</b></td>
 <td><input name="Valore[<?php echo $id; ?>]"></td>
</tr>
<?php
 }
?>
<tr>
    <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

In questo caso abbiamo come risultato un vettore contenente le coppie (indice, valore). Di seguito il codice per visualizzare il risultato del FORM.

```
<table cellspacing="2" cellpadding="5" border="0">
<?php
$Valore = &$_POST['Valore'];
$query = "SELECT * FROM campi_form ORDER BY etichetta";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
    $etichetta = htmlspecialchars ($row['etichetta']);
    $id = $row['id'];
?>
<tr>
 <td><b><?php echo $etichetta; ?>:</b></td>
 <td><?php echo $Valore[$id]; ?></td>
</tr>
<?php
}
?>
</table>
```

Si noti che per avere tutta la lista degli indici occorre effettuare una query, in modo da stampare anche le etichette dove i valori non stati immessi.

Con la tecnica precedente è possibile risolvere due problemi:

- aggiornare contemporaneamente più record di una tabella.
- implementare delle anagrafiche aventi un numero di campi indeterminato;

Aggiornare contemporaneamente più record

Ipotizziamo che all'interno di un database, abbiamo una tabella con pochi record, e vogliamo implementare un meccanismo che ci permette di modificarli con un singolo FORM. Questo è possibile mediante l'implementazione di un FORM dinamico.
Consideriamo una tabella avente la seguente struttura

```
CREATE TABLE `colori` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
```

```
`nome` VARCHAR( 50 ) NOT NULL ,
`html` VARCHAR( 6 ) NOT NULL
)
```

Quindi vediamo come implementare la modifica di tutti i record della tabella con un FORM

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="si">
<table cellspacing="2" cellpadding="5" border="0">
<?php
  $query = "SELECT * FROM colori ORDER BY nome";
  $res = mysql_query($query) or errore_db($query);
  while ($row = mysql_fetch_array($res))
  {
     $nome = htmlspecialchars ($row['nome']);
     $html = htmlspecialchars ($row['html']);
     $id = $row['id'];
?>
<tr>
  <td><input size="10" name="nome[<?php echo $id; ?>]" value="<?php echo $nome; ?>"></td>
  <td><input size="6" maxlength="6" name="html[<?php echo $id; ?>]" value="<?php echo $html; ?>"></td>
</tr>
<?php
  }
?>
<tr>
     <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

Infine vediamo come implementare l'aggiornamento dei record della tabella

```
   $nomev = &$_POST['nome'];
   $htmlv = &$_POST['html'];
   $nome = &$_POST_SQL['nome'];
   $html = &$_POST_SQL['html'];
   $query = "SELECT * FROM colori";
   $res = mysql_query($query) or die (mysql_error());
   while ($row = mysql_fetch_array($res))
   {
     $id = $row['id'];
     if ( ($row['nome']!=$nomev[$id]) || ($row['html']!=$htmlv[$id]) )
     {
        $query = " UPDATE colori
                      SET nome = '".$nome[$id]."',
                            html = '".$html[$id]."'
                 WHERE id='$id'";
        mysql_query($query) or die (mysql_error());
     }
   }
```

Si noti che la query di aggiornamento viene eseguita solo se i valori dei campi sono realmente stati modificati.

Anagrafiche con un numero di campi indeterminato

Ipotizziamo di avere di dover gestire una anagrafica tramite una tabella; ipotizziamo in oltre di offrire la possibilità all'utente di poter aggiungere altri campi in anagrafica. L'unico modo è quello di disporre di due tabelle aggiuntive, di cui una archivia le etichette dei campi aggiuntivi, mentre l'altra archivia i valori associati alle etichette. Vediamo la struttura delle tre tabelle

```
CREATE TABLE `anagrafica` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
`nome` VARCHAR( 255 ) NOT NULL ,
`cognome` VARCHAR( 255 ) NOT NULL
)

CREATE TABLE `campi_anagrafica` (
`id` INT NOT NULL AUTO_INCREMENT PRIMARY KEY,
`nome` VARCHAR( 50 ) NOT NULL
)


CREATE TABLE `anagrafica_valori` (
  `id` int(11) NOT NULL auto_increment,
  `id_anagrafica` int(11) NOT NULL,
  `id_campo` int(11) NOT NULL,
  `valore` varchar(255) NOT NULL,
  PRIMARY KEY  (`id`),
  KEY `id_anagrafica` (`id_anagrafica`)
)
```

All'interno del FORM di inserimento/modifica dell'anagrafica avremo ai campi statici si affiancheranno i campi dinamici. Vediamo di seguito la porzione della pagina relativa al FORM

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="si">
<input type="hidden" name="id" value="<?php echo $_GET['id']; ?>">
<table width="500" cellspacing="0" cellpadding="0" border="0"    style="border: 1px solid #aaaaaa;">
<tr>
    <td width="250" valige="top">
    <table width="100%" cellspacing="5" cellpadding="2" border="0">
    <tr>
       <td>Nome</td>
       <td><input name="nome" value="<?php echo $row['nome']; ?>"></td>
    </tr>
    <tr>
       <td>Cognome</td>
       <td><input name="cognome" value="<?php echo $row['cognome']; ?>"></td>
   </tr>
<?php
 $query = " SELECT c.id, c.nome, v.valore
            FROM campi_anagrafica AS c
            LEFT JOIN anagrafica_valori AS v ON v.id_anagrafica='$_GET_SQL[id]' AND v.id_campo=c.id";
 $res = mysql_query($query) or errore_db($query);
 while ($row = mysql_fetch_array($res))
 {
    $nome = htmlspecialchars ($row['nome']);
    $valore = htmlspecialchars ($row['valore']);
    $id_campo = $row['id'];
```

```
?>
    <tr>
       <td><?=$nome?></td>
       <td><input name="valore[<?=$id_campo?>]" value="<?=$valore?>"></td>
    </tr>
<?php
  }
?>
<tr>
    <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

Le query di inserimento e modifica della tabella anagrafica sono simili a quelle proposta nell'esempio del terzo capitolo. Di seguito il codice per l'inserimento

```
$sql = " INSERT INTO anagrafica (nome, cognome)
          VALUES ('$_POST_SQL[nome]', '$_POST_SQL[cognome]') ";
mysql_query($sql) or errore_db($query);

$id = mysql_insert_id();
```

Si noti che per l'operazione di inserimento dei campi dinamici abbiamo bisogno della chiave del record inserito.

Di seguito il codice per la modifica

```
$sql = " UPDATE rubrica
          SET  nome = '$_POST_SQL[nome]',
                cognome  = '$_POST_SQL[cognome]'
          WHERE id = '$_POST_SQL[id]' ";
mysql_query($sql) or errore_db($sql);

$id = $_POST_SQL['id'];
```

I codici per la modifica e l'inserimento de valori dei campi dinamici sono equivalenti, quindi è possibile utilizzare il codice seguente in entrambi i casi

```
 $valorev = &$_POST['valore'];
 $valore = &$_POST_SQL['valore'];

 $query = " SELECT c.id, v.valore, v.id_campo
            FROM campi_anagrafica AS c
            LEFT JOIN anagrafica_valori AS v ON v.id_anagrafica='$id' AND v.id_campo=c.id ";
 $res = mysql_query($query) or errore_db($query);
 while ($row = mysql_fetch_array($res))
 {
    $id_campo = $row['id'];
    if ($valorev[$id_campo] != $row['valore'])
    {
          $valore_form = $valore[$id_campo];
          if ($row['id_campo'])
          {
                $sql = " UPDATE anagrafica_valori
                          SET valore = '$valore_form'
                          WHERE id_anagrafica = '$id' AND id_campo = '$id_campo' ";
```

```
        }
        else
        {
                $sql = " INSERT INTO anagrafica_valori (id_anagrafica, id_campo, valore)
                        VALUES ('$id', '$id_campo', '$valore_form')  ";
        }
        mysql_query($sql) or errore_db($sql);
    }
 }
```

Se al momento della creazione non vengono inseriti dei valori, allora i record relativi ai valori dei campi dinamici non vengono creati. Se un valore di un campo è differente dal valore precedente allora avviene l'operazione di modifica o inserimento del record a seconda se il record era stato precedentemente inserito. Come vedremo più avanti, in questo modo possiamo agevolmente gestire l'inserimento di nuovi campi dinamici.
Per l'eliminazione di un record della tabella anagrafica, occorre eliminare anche i record associati nella tabella anagrafica_valori. Di seguito il codice.

```
If ($_POST['id_elimina'])
{
    $sql="DELETE FROM anagrafica_valori WHERE id_anagrafica = '$_POST_SQL[id_elimina]' ";
    mysql_query($sql) or errore_db($sql);

    $sql="DELETE FROM anagrafica WHERE id = '$_POST_SQL[id_elimina]' ";
    mysql_query($sql) or errore_db($sql);
}
```

Gestione dei campi dinamici

Dopo avere spiegato come dare la possibilità di gestire una anagrafica con campi dinamici, occorre far vedere come gestire l'inserimento di nuovi campi o l'eliminazione di campi esistenti. Sono infatti queste due operazioni che rendono questi campi veramente dinamici.
Grazie alla tecnica di aggiornare i campi vista in precedenza, aggiungere un nuovo campo è una operazione piuttosto semplice. Di seguito il codice

```
if ($_POST['azione']=='add')
{
   $sql=" INSERT INTO campi_anagrafica (nome) VALUES ('$_POST[nome]') ";
   mysql_query($sql) or errore_db($sql);

   header("Location: $_SERVER[PHP_SELF]");
}
```

È facile intuire che il codice appena visto, è facilmente inseribile nel solito schema visto in precedenza di inserimento e modifica dei dati.
Per l'eliminazione occorre prima togliere tutti i riferimenti dalla tabella dei valori. Di seguito il codice

```
 If ($_GET['id_elimina'])
{
    $sql="DELETE FROM anagrafica_valori WHERE id_campo = '$_GET[id_elimina]' ";
    mysql_query($sql) or errore_db($sql);

    $sql="DELETE FROM campi_anagrafica WHERE id = '$_GET[id_elimina]' ";
    mysql_query($sql) or errore_db($sql);

    header("Location: $_SERVER[PHP_SELF]");
}
```

Sempre come visto in precedenza, il codice di eliminazione appena visto è facilmente inseribile nella schema di visualizzazione dei dati; in questo caso visualizzazione dei campi dinamici.

Ricerche tramite campi dinamici

In pratica, la creazione del FORM di ricerca dei campi dinamici è simile al FORM di inserimento e modifica. Ovviamente poiché il FORM è situato al di sopra della tabella di visualizzazione, si pone un semplice problema: i campi dinamici sono un numero indefinito. Occorre quindi limitare la dimensione del FORM tramite un <DIV> con proprietà di scorrimento. Poiché i campi input devono essere allineati in due colonne, occorre un metodo per suddividere la lista dei campi dinamici. In pratica ottenendo due vettori è possibile gestire agevolmente le due visualizzazioni (una per colonna).

```
<?php
$i=1;
$query = " SELECT * FROM campi_anagrafica ";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
    $id_campo = $row['id'];
    $nome = htmlspecialchars ($row['nome']);
    if (($i/2)==(int)($i/2)) $campi_destra[$id_campo]=$nome;
    else $campi_sinistra[$id_campo]=$nome;
    $i++;
}
$valore = &$_POST['valore'];
?>
<form style="margin:5px" name="filtri" action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="id_elimina" value="">
<div style="overflow:auto;border: 1px solid #aaaaaa; width:720px;height:110px;">
<table width="700" cellspacing="0" cellpadding="0" border="0">
<tr>
    <td width="350" valign="top">
     <table width="100%" cellspacing="5" cellpadding="2" border="0">
     <tr>
          <td>Nome</td>
          <td><input name="nome" value="<?php echo $_POST['nome']; ?>"></td>
     </tr>
<?php
 foreach($campi_sinistra as $id_campo => $nome)
 {
?>
     <tr>
          <td><?=$nome?></td>
          <td><input name="valore[<?=$id_campo?>]" value="<?php echo $valore[$id_campo]; ?>"></td>
     </tr><?php
 }
?>
     </table>
    </td>
    <td width="350" valign="top">
     <table width="100%" cellspacing="5" cellpadding="2" border="0">
     <tr>
          <td>Cognome</td>
          <td><input name="cognome" value="<?php echo $_POST['cognome']; ?>"></td>
     </tr>
```

```
<?php
 foreach($campi_destra as $id_campo => $nome)
 {
?>
      <tr>
            <td><?=$nome?></td>
            <td><input name="valore[<?=$id_campo?>]" value="<?php echo $valore[$id_campo]; ?>"></td>
      </tr><?php
 }
?>
      </table>
    </td>
</tr>
</table>
</div>
……
</form>
```

A questo punto manca solamente l'implementazione dei filtri per la visualizzazione. Per i campi della tabella anagrafica non ci sono problemi

```
$sql_add = "";
if ($_POST_SQL['nome'])          $sql_add .= " AND r.nome LIKE '%$_POST_SQL[nome]%' ";
if ($_POST_SQL['cognome'])       $sql_add .= " AND r.cognome LIKE '%$_POST_SQL[cognome]%' ";
```

Poiché i campi dinamici non sono definiti nella tabella anagrafica, non è possibile utilizzare la stessa tecnica vista in precedenza sui filtri. La soluzione è quella di utilizzare gli operatori INNER JOIN sulla tabella anagrafica_valori. Di seguito le modifiche da apportare al codice

```
$valore_sql = &$_POST_SQL['valore'];
$i=1;
$query = " SELECT * FROM campi_anagrafica ";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
   $id_campo = $row['id'];
   $valore_campo=$valore_sql[$id_campo];
   if ($valore_campo)
    $query_campi .= "
        INNER JOIN anagrafica_valori AS v$id_campo ON v$id_campo.id_anagrafica = a.id
                AND v$id_campo.id_campo = '$id_campo'
                AND v$id_campo.valore LIKE '%$valore_campo%' ";

   $nome = htmlspecialchars ($row['nome']);
   if (($i/2)==(int)($i/2)) $campi_destra[$id_campo]=$nome;
   else $campi_sinistra[$id_campo]=$nome;
   $i++;
}
```

Tipi di campi dinamici

Finora tutti i campi dinamici erano dei semplici campi testo, e per ognuno di essi veniva indicato solo l'etichetta. Adesso vedremo come specificare il tipo di campo da utilizzare. In pratica si tratta di modificare la tabella campi_anagrafica, aggiungendo un campo che conterrà un numero che identifica il tipo di campo da utilizzare. Le diverse tipologie possono ovviamente comportare una implementazione più complessa del

programma, ad esempio se si specifica che il tipo [3] corrisponde ad un numero di telefono, nel FORM di inserimento occorre implementare un codice dinamico javascript in grado di verificare che tutti i campi definiti [3] siano effettivamente numeri di telefono. Un altro esempio e quando si specifica che un certo numero corrisponda al tipo virgola mobile.
Quando si vuole specificare che un determinato numero corrisponda ad una <SELECT> ed abbiamo a che fare con un numero limitato di voci, possiamo aggiungere dei campi aggiuntivi nella tabella campi_anagrafica. In questo modo è l'utente che specifica le voci da inserire. Invece se il numero corrisponde ad una <SELECT> con valori provenienti da una query, l'utente è vincolato dalla query definita dall'implementazione del programma.
Nel successivo esempio vedremo i principi di base per implementare la definizione di più tipi di dati che i nostri campi possono assumere, in particolare:

- Testo semplice;
- Opzione (checkbox);
- Numero di telefono;
- <SELECT> con (massimo 6) voci;

di seguito la tabella campi_anagrafica aggiornata

```
CREATE TABLE `campi_anagrafica` (
  `id` int(11) NOT NULL auto_increment,
  `nome` varchar(50) NOT NULL,
  `tipo` int(11) NOT NULL default '1',
  `voce1` varchar(50) NOT NULL,
  `voce2` varchar(50) NOT NULL,
  `voce3` varchar(50) NOT NULL,
  `voce4` varchar(50) NOT NULL,
  `voce5` varchar(50) NOT NULL,
  `voce6` varchar(50) NOT NULL,
  PRIMARY KEY  (`id`)
)
```

Il FORM di inserimento dei campi è piuttosto semplice

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="add">
<table cellspacing="2" cellpadding="5" border="0">
<tr>
  <td>Etichetta</td>
  <td><input size="10" maxlength="50" name="nome"></td>
  <td>Voce 1</td>
  <td><input size="10" maxlength="50" name="voce1"></td>
  <td>Voce 2</td>
  <td><input size="10" maxlength="50" name="voce2"></td>
  <td>Voce 3</td>
  <td><input size="10" maxlength="50" name="voce3"></td>
</tr>
<tr>
  <td>Tipo</td>
  <td><select name="tipo">
  <option value="1">Testo</option>
  <option value="2">Opzione</option>
  <option value="3">Telefono</option>
  <option value="4">Select</option>
     </select></td>
  <td>Voce 4</td>
```

```
  <td><input size="10" maxlength="50" name="voce4"></td>
  <td>Voce 5</td>
  <td><input size="10" maxlength="50" name="voce5"></td>
  <td>Voce 6</td>
  <td><input size="10" maxlength="50" name="voce6"></td>
</tr>
<tr>
  <td align="center" colspan="2"><input type="submit"></td>
</tr>
</table>
</form>
```

Di seguito il codice per eseguire la query di inserimento

```
if ($_POST['azione']=='add')
{
   $sql=" INSERT INTO campi_anagrafica
         (nome, tipo, voce1, voce2, voce3, voce4, voce5, voce6)
         VALUES
         ('$_POST[nome]', '$_POST[tipo]', '$_POST[voce1]', '$_POST[voce2]',
          '$_POST[voce3]', '$_POST[voce4]', '$_POST[voce5]', '$_POST[voce6]') ";
   mysql_query($sql) or errore_db($sql);

   header("Location: $_SERVER[PHP_SELF]");
}
```

Per la modifica di campi conviene utilizzare la tecnica vista in precedenza che permette di modificare più record contemporaneamente

```
<form action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="azione" value="si">
<table cellspacing="2" cellpadding="5" border="0">
<tr>
    <td><b>Etichetta</b></td>
    <td><b>Tipo</b></td>
    <td><b>Voce 1</b></td>
    <td><b>Voce 2</b></td>
    <td><b>Voce 3</b></td>
    <td><b>Voce 4</b></td>
    <td><b>Voce 5</b></td>
    <td><b>Voce 6</b></td>
    <td><b>Azioni</b></td>
</tr>
<?php
 $query = "SELECT * FROM campi_anagrafica ORDER BY nome";
 $res = mysql_query($query) or die (mysql_error());
 while ($row = mysql_fetch_array($res))
 {
    $nome = htmlspecialchars($row['nome']);
    $id = $row['id'];
    $tipo = $row['tipo'];
    $voce1 = htmlspecialchars($row['voce1']);
    $voce2 = htmlspecialchars($row['voce2']);
    $voce3 = htmlspecialchars($row['voce3']);
    $voce4 = htmlspecialchars($row['voce4']);
```

```
    $voce5 = htmlspecialchars($row['voce5']);
    $voce6 = htmlspecialchars($row['voce6']);
?>
<tr>
   <td><input size="10" maxlength="50"
               name="nome[<?php echo $id; ?>]" value="<?php echo $nome; ?>"></td>
   <td><select name="tipo[<?php echo $id; ?>]">
 <option value="1">Testo</option>
 <option <?php if ($tipo==2) echo "selected"; ?> value="2">Opzione</option>
 <option <?php if ($tipo==3) echo "selected"; ?> value="3">Telefono</option>
 <option <?php if ($tipo==4) echo "selected"; ?> value="4">Select</option>
    </select></td>
   <td><input size="10" maxlength="50"
               name="voce1[<?php echo $id; ?>]" value="<?php echo $voce1; ?>"></td>
   <td><input size="10" maxlength="50"
               name="voce2[<?php echo $id; ?>]" value="<?php echo $voce2; ?>"></td>
   <td><input size="10" maxlength="50"
               name="voce3[<?php echo $id; ?>]" value="<?php echo $voce3; ?>"></td>
   <td><input size="10" maxlength="50"
               name="voce4[<?php echo $id; ?>]" value="<?php echo $voce4; ?>"></td>
   <td><input size="10" maxlength="50"
               name="voce5[<?php echo $id; ?>]" value="<?php echo $voce5 ?>"></td>
   <td><input size="10" maxlength="50"
               name="voce6[<?php echo $id; ?>]" value="<?php echo $voce6; ?>"></td>
   <td><a href="lista_campi.php?id_elimina=<?php echo $id; ?>"
               onClick="javascript: return confirm('Sei sicuro di voler eliminare il campo?')"
               ><img src="<?=url_principale?>img/elimina.gif" width="16" height="16" border="0"
                     title="Elimina il campo"></a></td>
</tr>
<?php
 }
?>
<tr>
   <td colspan="2" align="center"><input type="submit"></td>
</tr>
</table>
</form>
```

Di seguito il codice per eseguire la query di modifica

```
if ($_POST['azione']=='si')
{
   $nome = &$_POST['nome'];
   $tipo = &$_POST['tipo'];
   $voce1 = &$_POST['voce1'];
   $voce2 = &$_POST['voce2'];
   $voce3 = &$_POST['voce3'];
   $voce4 = &$_POST['voce4'];
   $voce5 = &$_POST['voce5'];
   $voce6 = &$_POST['voce6'];
   $query = "SELECT * FROM campi_anagrafica ";
   $res = mysql_query($query) or die (mysql_error());
   while ($row = mysql_fetch_array($res))
   {
```

```
    $id = $row['id'];
    $query = " UPDATE campi_anagrafica
            SET nome='".$nome[$id]."',
               tipo='".$tipo[$id]."',
               voce1='".$voce1[$id]."',
               voce2='".$voce2[$id]."',
               voce3='".$voce3[$id]."',
               voce4='".$voce4[$id]."',
               voce5='".$voce5[$id]."',
               voce6='".$voce6[$id]."'
            WHERE id='$id'";
     mysql_query($query) or die (mysql_error());
   }
   header("Location: $_SERVER[PHP_SELF]");
}
```

Mentre il codice relativo al FORM di inserimento/modifica dell'anagrafica visto precedentemente necessita di alcune modifiche, il codice delle query di inserimento e modifica rimangono invariate. Vediamo di seguito la porzione della pagina relativa al FORM

```
<form onSubmit="javascript: return convalidaForm(this)" action="<?php echo $_SERVER['PHP_SELF']; ?>"
method="post">
<input type="hidden" name="azione" value="si">
<input type="hidden" name="id" value="<?php echo $_GET['id']; ?>">
<table width="400" cellspacing="2" cellpadding="2" border="0"    style="border: 1px solid #aaaaaa;">
<tr>
       <td>Nome</td>
       <td><input name="nome" value="<?php echo $row['nome']; ?>"></td>
</tr>
<tr>
        <td>Cognome</td>
        <td><input name="cognome" value="<?php echo $row['cognome']; ?>"></td>
</tr><?php
$query = " SELECT c.*, v.valore
         FROM campi_anagrafica AS c
         LEFT JOIN anagrafica_valori AS v ON v.id_anagrafica='$_GET_SQL[id]'
         AND v.id_campo=c.id";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
   $nome = htmlspecialchars ($row['nome']);
   $valore = htmlspecialchars ($row['valore']);
   $voce1 = htmlspecialchars ($row['voce1']);
   $voce2 = htmlspecialchars ($row['voce2']);
   $voce3 = htmlspecialchars ($row['voce3']);
   $voce4 = htmlspecialchars ($row['voce4']);
   $voce5 = htmlspecialchars ($row['voce5']);
   $voce6 = htmlspecialchars ($row['voce6']);
   $id_campo = $row['id'];
?>
<tr>
        <td><?=$nome?></td>
        <td><?php
        if ($row['tipo']==1)
```

```
        { ?><input name="valore[<?=$id_campo?>]" value="<?=$valore?>"><?php }
        if ($row['tipo']==2)
        { ?><input name="valore[<?=$id_campo?>]" <?php if ($valore) echo "checked"; ?> type="checkbox"
                    value="1"><?php }
        if ($row['tipo']==3)
        { ?><input name="valore[<?=$id_campo?>]" value="<?=$valore?>"><?php }
        if ($row['tipo']==4)
        { ?><select name="valore[<?=$id_campo?>]">
           <option></option>
           <option <?php if ($valore==1) echo "selected";?> value="1"><?=$voce1?></option>
           <option <?php if ($valore==2) echo "selected";?> value="2"><?=$voce2?></option>
           <option <?php if ($valore==3) echo "selected";?> value="3"><?=$voce3?></option>
           <option <?php if ($valore==4) echo "selected";?> value="4"><?=$voce4?></option>
           <option <?php if ($valore==5) echo "selected";?> value="5"><?=$voce5?></option>
           <option <?php if ($valore==6) echo "selected";?> value="6"><?=$voce6?></option>
           </select><?php }
        ?></td>
</tr><?php
}
?>
<tr>
        <td align="center" colspan="2"><input type="submit">
        </td>
</tr>
</table>
</form>
```

Nel seguente codice vedremo le modifiche necessarie per gestire la funzione Javascript di convalida del FORM

```
<script type="text/javascript">
function convalidaForm(form)
{
……….
<?php
$query = " SELECT c.*
         FROM campi_anagrafica AS c ";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
   if ($row['tipo']==3)
   {
   ?>
  if (!VerificaTelefono(form.elements("valore[<?=$row['id']?>]").value))
  {
     errori += "\n Il campo <?=$row['nome']?> non è un numero di telefono valido";
     if (!primo_campo_errato) primo_campo_errato = form.elements("valore[<?=$row['id']?>]");
  }
   <?php
   }
}
?>
……
```

```
  if (errori != "")
  {
    alert("Attenzione:" + errori);
    primo_campo_errato.focus();
    return false;
  }
}
</script>
```

Anche al codice per il FORM di ricerca necessita qualche modifica. Di seguito il calcolo dei filtri per la query e la creazione dei dati delle due colonne del FORM

```
$valore_sql = &$_POST_SQL['valore'];

$i=1;
$query = " SELECT * FROM campi_anagrafica ";
$res = mysql_query($query) or errore_db($query);
while ($row = mysql_fetch_array($res))
{
   $id_campo = $row['id'];
   $valore_campo=$valore_sql[$id_campo];
   if ($valore_campo)
   {
    $tipo = $row['tipo'];
    $query_campi .= "
         INNER JOIN anagrafica_valori AS v$id_campo ON v$id_campo.id_anagrafica = a.id
                  AND v$id_campo.id_campo = '$id_campo' ";

    if (($tipo==1) || ($tipo==3) )
       $query_campi .= " AND v$id_campo.valore LIKE '%$valore_campo%' ";
    else $query_campi .= " AND v$id_campo.valore = '%$valore_campo%' ";

   }

   $nome = htmlspecialchars ($row['nome']);
   if (($i/2)==(int)($i/2))
   {
     $campi_destra[$id_campo]['nome']=$nome;
     $campi_destra[$id_campo]['tipo']=$row['tipo'];
     $campi_destra[$id_campo]['voce1']=$row['voce1'];
     $campi_destra[$id_campo]['voce2']=$row['voce2'];
     $campi_destra[$id_campo]['voce3']=$row['voce3'];
     $campi_destra[$id_campo]['voce4']=$row['voce4'];
     $campi_destra[$id_campo]['voce5']=$row['voce5'];
     $campi_destra[$id_campo]['voce6']=$row['voce6'];
   }
   else
   {
     $campi_sinistra[$id_campo]['nome']=$nome;
     $campi_sinistra[$id_campo]['tipo']=$row['tipo'];
     $campi_sinistra[$id_campo]['voce1']=$row['voce1'];
     $campi_sinistra[$id_campo]['voce2']=$row['voce2'];
     $campi_sinistra[$id_campo]['voce3']=$row['voce3'];
     $campi_sinistra[$id_campo]['voce4']=$row['voce4'];
```

```
    $campi_sinistra[$id_campo]['voce5']=$row['voce5'];
    $campi_sinistra[$id_campo]['voce6']=$row['voce6'];
  }
  $i++;
}
```

Nel calcolo dei filtri, si noti che in base al tipo viene applicato l'operatore LIKE, o l'operatore di uguaglianza. Vediamo di seguito il codice relativo al FORM.

```
<form style="margin:5px" name="filtri" action="<?php echo $_SERVER['PHP_SELF']; ?>" method="post">
<input type="hidden" name="id_elimina" value="">
<div style="overflow:auto;border: 1px solid #aaaaaa; width:720px;height:110px;">
<table width="700" cellspacing="0" cellpadding="0" border="0">
<tr>
    <td width="350" valign="top">
     <table width="100%" cellspacing="5" cellpadding="2" border="0">
     <tr>
            <td>Nome</td>
            <td><input name="nome" value="<?php echo $_POST['nome']; ?>"></td>
     </tr>
<?php
 foreach($campi_sinistra as $id_campo => $vett)
 {
    $nome=$vett['nome'];
    $voce1=$vett['voce1'];
    $voce2=$vett['voce2'];
    $voce3=$vett['voce3'];
    $voce4=$vett['voce4'];
    $voce5=$vett['voce5'];
    $voce6=$vett['voce6'];
    $val=$valore[$id_campo];
?>
     <tr>
            <td><?=$nome?></td>
            <td><?php
      if ($vett['tipo']==1)
      { ?><input name="valore[<?=$id_campo?>]" value="<?=$val?>"><?php }
      if ($vett['tipo']==2)
      { ?><input name="valore[<?=$id_campo?>]" <?php if ($val) echo "checked"; ?> type="checkbox"
                value="1"><?php }
      if ($vett['tipo']==3)
      { ?><input name="valore[<?=$id_campo?>]" value="<?=$val?>"><?php }
      if ($vett['tipo']==4)
      { ?><select name="valore[<?=$id_campo?>]">
        <option></option>
        <option <?php if ($val==1) echo "selected";?> value="1"><?=$voce1?></option>
        <option <?php if ($val==2) echo "selected";?> value="2"><?=$voce2?></option>
        <option <?php if ($val==3) echo "selected";?> value="3"><?=$voce3?></option>
        <option <?php if ($val==4) echo "selected";?> value="4"><?=$voce4?></option>
        <option <?php if ($val==5) echo "selected";?> value="5"><?=$voce5?></option>
        <option <?php if ($val==6) echo "selected";?> value="6"><?=$voce6?></option>
        </select><?php }
      ?></td>
    </tr><?php
 }
```

```
?>
      </table>
     </td>
     <td width="350" valign="top">
      <table width="100%" cellspacing="5" cellpadding="2" border="0">
      <tr>
                <td>Cognome</td>
                <td><input name="cognome" value="<?php echo $_POST['cognome']; ?>"></td>
      </tr>
<?php
  foreach($campi_destra as $id_campo => $vett)
  {
     $nome=$vett['nome'];
     $voce1=$vett['voce1'];
     $voce2=$vett['voce2'];
     $voce3=$vett['voce3'];
     $voce4=$vett['voce4'];
     $voce5=$vett['voce5'];
     $voce6=$vett['voce6'];
     $val=$valore[$id_campo];
?>
      <tr>
                <td><?=$nome?></td>
                <td><?php
        if ($vett['tipo']==1)
        { ?><input name="valore[<?=$id_campo?>]" value="<?=$val?>"><?php }
        if ($vett['tipo']==2)
        { ?><input name="valore[<?=$id_campo?>]" <?php if ($val) echo "checked"; ?> type="checkbox"
                    value="1"><?php }
        if ($vett['tipo']==3)
        { ?><input name="valore[<?=$id_campo?>]" value="<?=$val?>"><?php }
        if ($vett['tipo']==4)
        { ?><select name="valore[<?=$id_campo?>]">
           <option></option>
           <option <?php if ($val==1) echo "selected";?> value="1"><?=$voce1?></option>
           <option <?php if ($val==2) echo "selected";?> value="2"><?=$voce2?></option>
           <option <?php if ($val==3) echo "selected";?> value="3"><?=$voce3?></option>
           <option <?php if ($val==4) echo "selected";?> value="4"><?=$voce4?></option>
           <option <?php if ($val==5) echo "selected";?> value="5"><?=$voce5?></option>
           <option <?php if ($val==6) echo "selected";?> value="6"><?=$voce6?></option>
           </select><?php }
        ?></td>
      </tr><?php
  }
?>
      </table>
     </td>
</tr>
</table>
</div>
……..
</form>
```

Come esercizio, il lettore può aggiungere anche altri tipi, come il campo numerico in virgola mobile, il campo data, il campo E-Mail, la <SELECT> prefissata (ad esempio sulle province) ed altri personalizzati.

Box

Un box è un contenitore con cornice, titolo e contenuto.

```
<table style="border: solid 1px #aaaaaa" cellspacing="0" cellpadding="0" border="0" width="240">
<tr>
        <td align=center>Titolo</td>
</tr>
<tr>
        <td bgcolor="#aaaaaa" height="1" ></td>
</tr>
<tr>
        <td> Cotenuto     ……   </td>
</tr>
</table>
```

Come vedremo nei prossimi paragrafi l'implementazione di un box può cambiare in base alle necessità. In particolare, con l'aiuto di javascript vedremo come associare ad un box alcune funzionalità.

Box a scomparsa

È possibile aggiungere ad un box la funzionalità di nascondere o visualizzare il contenuto, con un click del mouse sul titolo. Poiché il titolo rimane sempre visibile, occorre gestire il box tramite due tabelle una dove compare il titolo e l'atro dove compare il contenuto.
Tramite l'evento onClick posto nella tabella del titolo, possiamo richiamare una apposita funzione javascript (MostraNascindiBox()) che gestisce questa funzionalità. Per avere la possibilità di gestire più box a scomparsa all'interno della nostra pagina PHP, occorre identificare ogni tabella contenitore, utilizzando la proprietà id. Quando ,viene richiamata la funzione javascript MostraNascindiBox(), occorre passare l'identificativo della tabella da gestire. Per migliorare l'estetica è stata aggiunta alla tabella un'immagine che identifica lo stato del contenuto (visibile o non visubile). Di seguito la porzione di codice della pagina PHP relativa al box.

```
 <table class="box_superiore" onClick="javascript: MostraNascondiBox('box')">
<tr>
   <td width="15"></td>
   <td align=center>Titolo</td>
   <td width="15" align="right">
       <img id="imgbox" src="<?=url_principale?>img/aperto.gif" width="13" height="13" border="0"
            align="absmiddle" title="Clicca per i dettagli">
   </td>
</tr>
</table>
<table class="box_contenuto" id="box">
<tr>
   <td>Contenuto del Box
      <br>.....
      <br>.....
      <br>.....
   </td>
</tr>
</table>
```

Le classi box_superiore e box_inferiore si riferiscono a fogli di stile che determinano specificatamente l'estetica del box.
La funzione MostraNascindiBox() può essere inserita in uno script .js.

File *box.js*

```
function MostraNascondiBox (nome_id_box)
```

```
{
	var id_box = document.getElementById(nome_id_box);
	var id_img = document.getElementById("img"+nome_id_box);

	if ( id_box.style.display == "none" )
	{
		id_box.style.display = "";
		if ( id_img != "" ) id_img.src = icona_chiuso;
	}
	else
	{
		id_box.style.display = "none";
		if ( id_img != "" ) id_img.src = icona_aperto;
	}
}
```

Le variabili icona_aperto e icona_chiuso, devono essere definite in questo modo

```
<script type="text/javascript">
var icona_aperto="<?=url_principale?>img/aperto.gif";
var icona_chiuso="<?=url_principale?>img/chiuso.gif";
</script>
```

Se i vari box risiedono nella maggior parte del sito, possiamo inserire queste istruzioni nella pagina top.php. Nel codice precedente abbiamo impostato inizialmente il box, come aperto. Vediamo come impostare inizialmente il box come chiuso.

```
<table class="box_superiore" onClick="javascript: MostraNascondiBox('box')">
<tr>
  <td width="15"></td>
  <td align=center>Titolo</td>
  <td width="15" align="right">
    <img id="imgbox" src="<?=url_principale?>img/chiuso.gif" width="13" height="13" border="0"
        align="absmiddle" title="Clicca per i dettagli">
  </td>
</tr>
</table>
<table class="box_contenuto" id="box" style="display:none;">
<tr>
  <td>Contenuto del Box
    <br>.....
    <br>.....
    <br>.....
  </td>
</tr>
</table>
```

Box trascinabili

Aggiungendo altri eventi del mouse è possibile anche spostare il box con il mouse.
La gestione dello spostamento
la classe Muovi che si occuperà di gestire il movimento del box
Poiché alcune istruzioni possono differire da Firefox conviene implementare degli oggetti che astraggono le funzioni incompatibili fra i vari browser. Questi oggetti utilizzeranno la variabile is_firefox per decidere quali istruzioni eseguire. Di seguito la definizione di questa variabile

```
is_firefox = /Firefox/i.test(navigator.userAgent);
```

Quando viene richiamata una funzione evento, viene passata alla funzione una variabile contenente alcune informazioni su dove si è verificato l'evento. La classe Target permette di gestire le informazioni contenute in questa variabile. In particolare è in grado di ricavare le coordinate assolute del mouse all'interno della pagina PHP (MouseX_Pagina() e MouseY_Pagina()).

```
Target = function(e)
{
   this.e=e;
   if (is_firefox) this.target = e.target ? e.target : e.srcElement;
   else this.target = event.target ? event.target : event.srcElement;
}

Target.prototype.MouseX_Pagina = function()
{
   if (is_firefox) return this.e.pageX;
   else return event.clientX + document.body.scrollLeft;
}

Target.prototype.MouseY_Pagina = function()
{
   if (is_firefox) return this.e.pageY;
   else return event.clientY + document.body.scrollTop;
}
```

La classe Elemento permette di leggere e modificare le coordinate del box.

```
Elemento = function(id)
{
   this.SetElemento(id);
}

Elemento.prototype.SetElemento = function(id)
{
   if (typeof (id)=='string') this.id = document.getElementById(id);
   else this.id = id;
}

Elemento.prototype.SetTop = function(top)
{
   if (top<0) top=0;
   if (is_firefox) this.id.style.top = top;
   else this.id.style.pixelTop = top;
}

Elemento.prototype.GetTop = function()
{
   if (is_firefox) return parseInt(this.id.style.top.replace('px', ''));
   else return this.id.style.pixelTop;
}

Elemento.prototype.SetLeft = function(left)
{
   if (left<0) left=0;
```

```
   if (is_firefox) this.id.style.left = left;
   else this.id.style.pixelLeft = left;
}

Elemento.prototype.GetLeft = function()
{
   if (is_firefox) return parseInt(this.id.style.left.replace('px', ''));
   else return this.id.style.pixelLeft;
}
```

Vediamo adesso come implementare la classe Muovi. Il funzionamento del trascinamento si basa sul controllo di tre eventi del mouse: tasto premuto (onMouseDown), movimento (onMouseMove) e tasto rilasciato (onMouseUp). La proprietà Muovi.box memorizza l'oggetto relativo al contenitore da spostare. La proprietà Muovi.over viene utilizzata per verificare se il mouse si trova al di sopra della barra del titolo ovvero l'area dove è possibile trascinare il contenitore. Quest'ultima viene controllata tramite gli eventi del mouse: puntatore è appena entrato (onMouseOver) e puntatore è appena uscito (onMouseOut).
File box.*js*

```
…….
…….
Muovi = function()
{
   Muovi.box = new Elemento(false);
   Muovi.over = false;
   document.onmousedown = Muovi.MD;
   document.onmousemove = Muovi.MM;
   document.onmouseup = Muovi.MU;
}
…….
…….
```

Si noti che al momento della creazione della classe, vengono aggiunti al documento gli eventi del mouse. Quando questi eventi vengono richiamati, occorre determinare il riferimento del contenitore (tag <DIV>). Poiché il tag che ha generato l'evento è all'interno del tag <DIV>, occorre sviluppare una funzione che cerca fra i tag padri, il corrispondente tag <DIV>. Si noti che la funzione è stata generalizza per trovare il tag di nome pNomeTag.

```
function CercaPadre(el, pNomeTag)
{
   if (el == null) return null;
   else
   {
      if (el.nodeType == 1 && el.tagName.toLowerCase() == pNomeTag.toLowerCase()) return el;
      else return CercaPadre(el.parentNode, pNomeTag);
   }
}
```

Nella barra del titolo, viene controllata la proprietà Muovi.over

```
onmouseover="Muovi.over=true;" onmouseout="Muovi.over=false;"
```

Quando viene premuto il pulsante sulla barra del titolo, viene generato l'evento onMouseDown, che quindi richiama la funzione membro Muovi.MD(e); si noti che se il mouse si trova al di sopra della barra del titolo, allora Mouvi.over è settato a true. Le proprietà Muovi.X e Muovi.Y, mantengono le informazioni sulle differenze di coordinate tra il contenitore e il mouse al momento del click del mouse. Si noti che Muovi.X e Muovi.Y sono le coordinate del Mouse all'intermo del contenitore. Queste variabili vengono utilizzate per calcolare quelle nuove a seguito del trascinamento del mouse.

```
Muovi.MD = function(e)
{
  if (Muovi.over)
  {
    var t = new Target(e);
    var tParent = CercaPadre(t.target, "div");
    Muovi.box.SetElemento(tParent.id.toString());
    Muovi.X = t.MouseX_Pagina()-Muovi.box.GetLeft();
    Muovi.Y = t.MouseY_Pagina()-Muovi.box.GetTop();
  }
}
```

Al passaggio del mouse, viene richiamato l'evento onMouseMove, che quindi richiama la funzione membro Muovi.MM(e); ovviamente se l'oggetto Muovi.box contiene il riferimento del contenitore, allora si può procedere allo spostamento. Le nuove coordinate del contenitore, vengono aggiornate in base alla differenza tra le vecchie coordinate relative del Mouse e le nuove coordinate assolute del mouse.

```
Muovi.MM = function(e)
{
 if (Muovi.box.id)
 {
    var t = new Target(e);
    Muovi.box.SetTop(t.MouseY_Pagina() - Muovi.Y);
    Muovi.box.SetLeft(t.MouseX_Pagina() - Muovi.X);
 }
}
```

Quando viene rilasciato il pulsante, viene richiamato l'evento onMouseUp, che quindi richiama la funzione membro Muovi.MU(e); in questo caso la proprietà Muovi.box viene annullata, poiché il contenitore non deve essere più spostato.

```
Muovi.MU = function()
{
 Muovi.box = null;
}
```

Di seguito le modifiche da apportare al box

```
……..
<script type="text/javascript">
new Muovi();
</script>
<div id="box_trascinabile" style="position:absolute ; left:100px; top:100px; ">
<table class="box_superiore">
<tr>
   <td style="cursor: move;" onmouseover="Muovi.over=true;" onmouseout="Muovi.over=false;" width="15"></td>
   <td style="cursor: move;" onmouseover="Muovi.over=true;" onmouseout="Muovi.over=false;"
       align=center>Titolo</td>
   <td width="15" align="right">
      <a href="javascript: MostraNascondiBox('box')"><img id="imgbox" src="<?=url_principale?>img/aperto.gif"
         width="13" height="13" border="0" align="absmiddle" title="Clicca per i dettagli"></a>
   </td>
</tr>
</table>
<table class="box_contenuto" id="box">
<tr>
   <td>Contenuto del Box
```

```
      <br>.....
      <br>.....
      <br>.....
   </td>
</tr>
</table>
</div>
…….
```

Ovviamente la chiusura del box non può essere fatta più mediante click sul titolo, in quanto quest'ultimo deve servire per lo spostamento.
Il codice precedente, ha un difetto con Internet Explorer 6. Quando una finestra trascinabile viene spostata sopra una <SELECT>, si verifica uno strano effetto: nonostante si imposta la proprietà z-index molto alta al contenitore, essa risulterà sempre al di sotto di essa. La soluzione è di aggiungere un <IFRAME> al di sotto del contenitore.

Di seguito le modifiche de box

```
……..
<script type="text/javascript">
new Muovi();
</script>
<div id="box_trascinabile" style="position:absolute; z-index:10000; left:100px; top:100px;">
<table class="box_superiore">
<tr>
   <td style="cursor: move;" onmouseover="Muovi.over=true;" onmouseout="Muovi.over=false;" width="15"></td>
   <td style="cursor: move;" onmouseover="Muovi.over=true;" onmouseout="Muovi.over=false;"
        align=center>Titolo</td>
   <td width="15" align="right">
       <a href="javascript: MostraNascondiBox('box')"><img id="imgbox" src="<?=url_principale?>img/aperto.gif"
           width="13" height="13" border="0" align="absmiddle" title="Clicca per i dettagli"></a>
   </td>
</tr>
</table>
<table class="box_contenuto" id="box">
<tr>
   <td>Contenuto del Box
      <br>.....
      <br>.....
      <br>.....
   </td>
</tr>
</table>
</div>
<script type="text/javascript">
if (is_ie)
{
 var el = new Elemento("box_trascinabile");
 el.id.innerHTML+='<iframe src="" frameborder="0" width="100%" height="'+ el.GetHeight()
     +'" style="top:0px;position:absolute;z-index:1;"></iframe>';
}
</script>
…….
```

La variabile is_ie viene definita in questo modo

```
is_ie = ( /msie/i.test(navigator.userAgent) &&
                    !/opera/i.test(navigator.userAgent) );
```

Mentre la funzione membro GetHeight viene aggiunta alla classe Elemento

```
Elemento.prototype.GetHeight = function()
{
   return this.id.offsetHeight;
}
```

Ovviamente la proprietà z-index dell'IFRAME deve essere più bassa della proprietà z-index delle due tabelle. Avremo quindi gli stili delle tabelle definite in questo modo

```
.box_superiore{
  font: 12px Tahoma;
  font-weight : bold;
  border: solid 1px #aaaaaa;
  background:#eeeeee;
  height:25;
  width:240px;
  margin:0;
  padding:0;
  position:relative;
  z-index:10;
}

.box_contenuto{
  font: 12px Tahoma;
  padding:2px;
  font-weight : normal;
  border-bottom: solid 1px #aaaaaa;
  border-left: solid 1px #aaaaaa;
  border-right: solid 1px #aaaaaa;
  width:240px;
  margin:0;
  padding:0;
  position:relative;
  z-index:10;
}
```

Alfa-Testing e generazione dei dati

Nel processo di alfa-testing di un gestionale, per verificare l'efficienza delle query, ottimizzare la scelta degli indici nelle tabelle e verificare il funzionamento generale del programma è necessario disporre di un database, con un elevato numero di dati ed il più realistico possibile. Riempire il database mediante inserimenti manuali, è una procedura troppo lenta, quindi occorre implementare un programma di generazione dati automatico. Per poter generare dei dati in modo equivalente alla realtà, è necessario disporre di appositi indici statistici, che possono essere reperiti dalle aziende che effettuano il beta-testing. Il programma di generazione dei dati, dovrà essere configurato in modo da rispecchiare questi indici statistici e quindi ottenere un database simile a quelli reali.
Il database che verrà preso in esame è quello definito nel secondo capitolo, che essendo molto semplice rispetto ad un gestionale reale, si riesce meglio a riportare e sottolineare i concetti fondamentali, e le soluzioni più importanti. Nell'operazione di inserimento dati è opportuno utilizzare le librerie del programma che si sta implementando, in modo da diminuire i costi implementativi. Nei successivi esempi invece vengono eseguite query dirette a database in quanto si riescono a capire meglio i meccanismi funzionali del procedimento. Il cuore del programma di generazione dati, è composto da una libreria di classi. Per una migliore scalabilità, esse non eseguono direttamente le query.

Generare dati casuali

Generare numeri causali

In generale, se un'azienda ha un certo numero di clienti, nell'arco di un determinato lasso di tempo una parte di loro effettuerà la maggior parte degli acquisti, una piccola parte non acquisterà più, e vi si aggiungeranno altri nuovi clienti. Mentre è abbastanza facile emulare la perdita o l'acquisizione dei clienti, emulare una distribuzione degli acquisti da parte dei clienti è un'operazione complessa in quanto nel caso reale solo una parte dei clienti acquistano di più. Se impostiamo il programma in modo che i clienti acquistino in modo causale, otteniamo nel lungo periodo una distribuzione omogenea, e questo non corrisponde a quello che vogliamo.
Un altro problema è quello di assegnare casualmente le scorte di sicurezza che possono avere una particolare distribuzione; ad esempio una piccola parte degli articoli possono essere ad alta scorta di sicurezza, è la rimanente parte inferire o addirittura non sono gestiti a magazzino. Una tecnica è quella di generare numeri in modo da ottenere una distribuzione non omogenea, ad esempio generarne la maggior parte in un determinato punto, che può essere il valore massimo, minimo o centrato. Queste distribuzioni devono ovviamente essere parametrizate da un valore che stabilisca con quale probabilità i numeri scelti siano vicini al punto scelto; più il parametro è alto e maggiore è la probabilità che il numero sia vicino ad esso.
Le funzioni proposte calcolano un numero causale a partire da un intervallo tra x_1 e x_2; le loro distribuzioni sono rispettivamente crescente, decrescente e centrata. Di seguito il codice.

```
function get_random_crescente($x1, $x2, $e)
{
   $base = pow($x2-$x1, $e);
   $numero =  mt_rand(0, $base);
   $numero = ceil(pow($numero, 1/$e));
   return ($x1+$numero);
}


function get_random_decrescente($x1, $x2, $e)
{
   $base = pow($x2-$x1, $e);
   $numero =  mt_rand(0, $base);
   $numero = ceil(pow($numero, 1/$e));
   return ($x2-$numero);
}

function get_random_media($x1, $x2, $e)
{
   $distanza = $x2-$x1;
   $media = $x1+($distanza/2);
   $numero =  mt_rand(0, $distanza);
   if ($numero<$distanza/2) return get_random_crescente($x1, $media, $e);
   else return get_random_decrescente($media, $x2, $e);
}
```

Per verificare in che modo sono distribuiti i numeri occorre calcolare le ripetizioni su una generazione molto alta di numeri. Di seguito il codice

```
<?php
function stampa_distribuzione($vet)
{
?>
<table>
<?php
ksort($vet);
reset($vet);
foreach($vet AS $chiave => $valore)
{
   echo "<tr><td>[$chiave]</td><td>$valore</td></tr>";
}
?>
</table>
<?php
}

?>
<table cellpadding="5" cellspacing="5">
<tr>
<td>
<?php
for ($i=0; $i<100000; $i++)
{
   $numero=get_random_crescente(1, 10, 2);
   $vet[$numero]++;
}
echo "<b>Distribuzione crescente</b>";
stampa_distribuzione($vet);
?>
</td>
<td>
<?php
unset($vet);
for ($i=0; $i<100000; $i++)
{
   $numero=get_random_media(1, 10, 2);
   $vet[$numero]++;
}
echo "<b>Distribuzione al centro</b>";
stampa_distribuzione($vet);
?>
</td>
<td>
<?php
unset($vet);
for ($i=0; $i<100000; $i++)
{
   $numero=get_random_decrescente(1, 10, 2);
   $vet[$numero]++;
}
```

```
echo "<b>Distribuzione decrescente</b>";
stampa_distribuzione($vet);
?>
</td>
</tr>
</table>
```

L'esempio mostra le tre distribuzioni di generazione di numeri da uno a dieci, e per ogni tabella vengono generate 100.000 numeri. Anche se il risultato cambia ad ogni esecuzione della pagina, le ripetizioni dei numeri mostrano sempre la stessa tendenza. Di seguito un esempio.

Distribuzione crescente		**Distribuzione centrale**		**Distribuzione decrescente**	
[1]	1306	[1]	2384	[1]	20787
[2]	1247	[2]	2314	[2]	18180
[3]	3735	[3]	7080	[3]	15940
[4]	6076	[4]	11752	[4]	13570
[5]	8577	[5]	26404	[5]	10644
[6]	10787	[6]	26195	[6]	8436
[7]	13369	[7]	11923	[7]	6233
[8]	15791	[8]	7209	[8]	3791
[9]	18544	[9]	2430	[9]	1219
[10]	20568	[10]	2309	[10]	1200

Generare nomi casuali

Per generare ad esempio le ragioni sociali dei clienti, possiamo anche prenderli a caso da un database di nomi, ma in mancanza dobbiamo generali in modo casuale. Occorre premettere che di un articolo, mentre il nome è senza spazi, la descrizione li può contenere. Per facilitare l'implementazione, il nome di un articolo è composta da una parola, e la sua descrizione da un insieme di parole staccate da spazi che chiameremo frase. Di seguito il codice per generare parole e frasi.

```
function get_parola($caratteri)
{
    for ($i=0; $i<$caratteri; $i++)
    {
        $num=rand(ord('a'), ord('z'));
        $result .= chr($num);
     }
     return $result;
}

function get_frase($caratteri)
{
    $result="";
    while(strlen($result)<$caratteri)
    {
        $diff = $caratteri-strlen($result);
        if ($diff<4) break;
        $num = $diff;
        if ($diff>10) $num = rand(1, $diff/2);
        $result .= " ". get_parola($num);
    }
    return trim($result);
```

```
}
```

Si noti i le parole generate solo altamente incomprensibili e difficili da ricordare, e per questo motivo in fase di testing conviene modificare i nomi dei clienti e fornitori che si vogliono utilizzare nei test.

Generare i codici di clienti, fornitori e articoli

Per la generazione dei codici occorre conoscere la tecnica utilizzata dalle aziende. Solitamente per i clienti sono del tipo CXXXX dove XXXX è un numero progressivo (es. C1345, C1346, …). Anche per i fornitori viene utilizzato lo stesso meccanismo con FXXXX. Per gli articoli ci sono due tecniche, la prima consiste nel solito metodo YYXXXX, dove YY è il codice di categoria e XXXXX è un numero progressivo; la seconda si basa sul costruire un codice "parlante", dove le lettere e i numeri che compongono il codice rappresentano una sorta di breve descrizione dell'articolo. In pratica nel secondo metodo mediante il codice è possibile identificare facilmente il tipo di articolo.
Di seguito la classe che implementa il codice progressivo.

```
class genera_codice
{
   protected $codice_iniziale;
   protected $progressivo;
   protected $lunghezza_codice;

   public function __construct($codice_iniziale, $progressivo=0, $lunghezza_codice=5)
   {
      $this->codice_iniziale = $codice_iniziale;
      $this->progressivo = $progressivo;
      $this->lunghezza_codice = $lunghezza_codice;
   }

   public function successivo()
   {
      $this->progressivo++;
      $result = $this->progressivo;
      while (strlen($result)<($this->lunghezza_codice-1)) $result = "0".$result;
      return $this->codice_iniziale.$result;
   }
}
```

Di seguito un esempio per vedere come utilizzare la classe genera_codice per generare i codici dei clienti

```
<b>Creazione codici dei clienti</b>
<table>
<?php
$genera_codice = new genera_codice('C');
for ($i=0; $i<20; $i++) echo "<tr><td>".$genera_codice->successivo()."</td></tr>";
?>
</table>
```

Di seguito il risultato

Creazione codici dei clienti

C0001
C0002
C0003
C0004
C0005
C0006
C0007
C0008
C0009
C0010
C0011
C0012
C0013
C0014
C0015
C0016
C0017
C0018
C0019
C0020

Per quanto riguarda la creazione di codici parlanti, solitamente il codice è composto in questo modo: i primi caratteri rappresentano il tipo di articolo, gli altri corrispondono alle sue caratteristiche fisiche come ad esempio tipo di materiale, lunghezza, larghezza, diametro, taglia o colore. Quindi per generare una sequenza di questo tipo di codici, basta effettuare una permutazione delle varie caratteristiche. Di seguito la classe che implementa il codice parlante.

```
Class genera_permutazione
{
   protected $codice_iniziale;
   protected $lista_caratteristiche = array();
   protected $contatore = array();

   public function __construct($codice_iniziale, &$lista_caratteristiche = array())
   {
     $this->codice_iniziale = $codice_iniziale;
     $this->lista_caratteristiche = $lista_caratteristiche;
     $n = count($this->lista_caratteristiche);
     for ($i=0; $i<$n; $i++) $this->contatore[$i] = 0;
   }

   public function successivo()
   {
     $result = $this->codice_iniziale;
     $n = count($this->lista_caratteristiche);
     if ($n==0) return $result;
     for ($i=0; $i<$n; $i++)
     {
         $result .= $this->lista_caratteristiche[$i][$this->contatore[$i]];
     }
     $i=$n-1;
     while($i>=0)
```

```
    {
        $this->contatore[$i]++;
        if ($this->contatore[$i]<count($this->lista_caratteristiche[$i])) break;
        $this->contatore[$i]=0;
        $i--;
    }
    return $result;
  }
}
```

Di seguito un esempio per vedere come utilizzare la classe genera_permutazione per generare i codici degli articoli

```
<b>Creazione codici degli articoli</b>
<table>
<?php
$lista_caratteristiche = array(
array('AAS', 'BCB', 'XXX', 'ZZA'),
array('12', '14', '16', '18'),
array('05', '10', '20')
);

$genera_codice = new genera_permutazione('A1', $lista_caratteristiche);
for ($i=0; $i<20; $i++) echo "<tr><td>".$genera_codice->successivo()."</td></tr>";
?>
</table>
```

Di seguito il risultato

Creazione codici degli articoli

A1AAS1205
A1AAS1210
A1AAS1220
A1AAS1405
A1AAS1410
A1AAS1420
A1AAS1605
A1AAS1610
A1AAS1620
A1AAS1805
A1AAS1810
A1AAS1820
A1BCB1205
A1BCB1210
A1BCB1220
A1BCB1405
A1BCB1410
A1BCB1420
A1BCB1605
A1BCB1610

Generare le anagrafiche

Generare gli articoli

La generazione degli articoli è relativamente semplice, e consiste nel riempire rispettivamente le tabelle articolo e categoria_merceologica. Occorrono quindi alcuni dati statistici:

- Numero di categorie merceologiche;
- Numero di articoli;
- Numero effettivo di articoli gestiti con il magazzino (cioè aventi una scorta di sicurezza);
- Metodologia della generazione dei codici;
- Metodologia della generazione delle scorte di sicurezza;

Di seguito la classe relativa alla generazione delle categorie merceologiche

```
class genera_categorie
{
   public $categorie = array();
   protected $id_categoria;
   protected $lunghezza_nome;

         public function __construct($lunghezza_nome=20)
         {
            $this->id_categoria=1;
            $this->lunghezza_nome=$lunghezza_nome;

   }

   public function genera($num)
   {
      for ($i=0; $i<$num; $i++)
      {
         $this->categorie[$this->id_categoria]['id']=$this->id_categoria;
         $this->categorie[$this->id_categoria]['nome']=get_frase($this->lunghezza_nome);
         $this->id_categoria++;
      }
   }
}
```

Questa classe memorizza i dati generati in un vettore associativo, in modo da poterli utilizzare nella creazione delle query e nella successiva generazione degli articoli.
Di seguito la classe relativa alla generazione degli articoli

```
class genera_articoli
{
   public $articoli = array();
   protected $id_articolo;
   protected $lunghezza_descrizione;
   protected $genera_scorta_sicurezza;

         public function __construct($genera_scorta_sicurezza, $lunghezza_descrizione=20)
         {
            $this->id_articolo=1;
            $this->lunghezza_descrizione=$lunghezza_descrizione;
            $this->genera_scorta_sicurezza=$genera_scorta_sicurezza;
```

```
   }

   public function genera($num, &$genera_codici, $id_categoria)
   {
      $funzione=$this->genera_scorta_sicurezza;
      for ($i=0; $i<$num; $i++)
      {
         $scorta_sicurezza=$funzione();
         $this->articoli[$this->id_articolo]['id']=$this->id_articolo;
         $this->articoli[$this->id_articolo]['codice']=$genera_codici->successivo();
         $this->articoli[$this->id_articolo]['descrizione']=get_frase($this->lunghezza_descrizione);
         $this->articoli[$this->id_articolo]['id_categoria']=$id_categoria;
         $this->articoli[$this->id_articolo]['scorta_sicurezza']=$scorta_sicurezza;
         $this->articoli[$this->id_articolo]['scorta_fisica']=$scorta_sicurezza;
         $this->id_articolo++;
      }
   }
}
```

Anche questa classe memorizza i dati generati in un vettore associativo. Più in avanti verrà utilizzato nella creazione degli ordini.
La variabile $genera_scorta_sicurezza contiene il riferimento alla funzione che genera la scorta di sicurezza. L'implementazione di quest'ultima dipende dalla metodologia che si vuole applicare nell'assegnare le scorte di sicurezza agli articoli. Il metodo genera(), deve essere chiamata per ogni singola categoria.
Di seguito un codice di esempio per generare gli articoli di una categoria

```
<b>Generazione articoli</b>
<table border="1" cellpadding="2" cellspacing="0">
  <tr><td><b>id</b></td>
  <td><b>codice</b></td>
  <td><b>descrizione</b></td>
  <td><b>id_categoria</b></td>
  <td><b>scorta_sicurezza</b></td>
  <td><b>scorta_fisica</b></td>
</tr>
<?php

function get_random_scorta_sicurezza()
{
   if (rand(1,3)<3) return 0;
   return get_random_decrescente(0, 500, 2);
}

$lista_caratteristiche = array(
array('AAS', 'BCB', 'XXX', 'ZZA'),
array('12', '14', '16', '18'),
array('05', '10', '20')
);

$genera_codice = new genera_permutazione('A1', $lista_caratteristiche);
$genera_articoli = new genera_articoli('get_random_scorta_sicurezza');
$genera_articoli->genera(20, $genera_codice, 1);
for ($i=1; $i<=20; $i++)
{
   echo "<tr><td>".$genera_articoli->articoli[$i]['id']."</td>";
```

```
   echo "<td>".$genera_articoli->articoli[$i]['codice']."</td>";
   echo "<td>".$genera_articoli->articoli[$i]['descrizione']."</td>";
   echo "<td>".$genera_articoli->articoli[$i]['id_categoria']."</td>";
   echo "<td>".$genera_articoli->articoli[$i]['scorta_sicurezza']."</td>";
   echo "<td>".$genera_articoli->articoli[$i]['scorta_fisica']."</td></tr>";
}
?>
</table>
```

Generare i clienti

La generazione dei clienti consiste nel riempire rispettivamente le tabelle cliente e categoria Occorrono i seguenti dati statistici:

- Numero di categorie dei clienti;
- Numero di clienti;
- Numero di clienti che si perdono e acquisiscono nell'arco di un anno;
- Metodologia della distribuzione delle vendite verso i clienti;

Di seguito la classe relativa alla generazione dei clienti

```
class genera_clienti
{
   public $clienti = array();
   protected $id_cliente;
   protected $lunghezza_ragione;
   protected $genera_volume_ordini;
   protected $genera_volume_articoli;
   protected $genera_codice;

         public function __construct($genera_volume_ordini, $genera_volume_articoli, $lunghezza_ragione=20)
         {
            $this->id_cliente=1;
            $this->genera_codice=new genera_codice('C');
            $this->lunghezza_ragione=$lunghezza_ragione;
            $this->genera_volume_ordini=$genera_volume_ordini;
            $this->genera_volume_articoli=$genera_volume_articoli;
   }

   public function genera($num, $num_cat)
   {
      $funzione_volume_ordini=$this->genera_volume_ordini;
      $funzione_volume_articoli=$this->genera_volume_articoli;
      for ($i=0; $i<$num; $i++)
      {
         $volume_ordini=$funzione_volume_ordini();
         $volume_articoli=$funzione_volume_articoli();
         $id_categoria=rand(1, $num_cat);
         $this->clienti[$this->id_cliente]['id']=$this->id_cliente;
         $this->clienti[$this->id_cliente]['codice']=$this->genera_codice->successivo();
         $this->clienti[$this->id_cliente]['ragione']=get_frase($this->lunghezza_ragione);
         $this->clienti[$this->id_cliente]['id_categoria']=$id_categoria;
         $this->clienti[$this->id_cliente]['ordini']=$volume_ordini;
         $this->clienti[$this->id_cliente]['articoli']=$volume_articoli;
         $this->id_cliente++;
      }
   }
```

```
}
```

La variabile $genera_volumi_ordini contiene il riferimento alla funzione che genera la media di ordini in un anno e la variabile $genera_volumi_articoli contiene la funzione che genera la media di articoli per ordine; queste informazioni non devono essere archiviate sul database del gestionale, ma servono per calcolare successivamente gli ordini. L'implementazione di queste due funzioni dipende dalla metodologia che si vuole applicare nell'assegnare i volumi di vendita complessivi ai clienti. Il metodo genera(), deve essere chiamata indicando il numero di clienti e il numero di categorie. Si noti che l'assegnazione delle categorie sui clienti parte dal presupposto che gli indici delle categorie sono numeri che partono da uno e seguono la progressione numerica (cioè 1,2,3, … senza salti).
Di seguito il codice per la creazione dei clienti

```
<b>Generazione clienti</b>
<table border="1" cellpadding="2" cellspacing="0">
  <tr><td><b>id</b></td>
  <td><b>codice</b></td>
  <td><b>ragione</b></td>
  <td><b>id_categoria</b></td>
  <td><b>Volume ordini</b></td>
  <td><b>Volume articoli</b></td>
</tr>
<?php

function get_random_volume_ordini()
{
   return get_random_media(0, 200, 2);
}

function get_random_volume_articoli()
{
   return get_random_decrescente(1, 20, 2);
}

$genera_clienti = new genera_clienti('get_random_volume_ordini', 'get_random_volume_articoli');
$genera_clienti->genera(20, 5);
for ($i=1; $i<=20; $i++)
{
   echo "<tr><td>".$genera_clienti->clienti[$i]['id']."</td>";
   echo "<td>".$genera_clienti->clienti[$i]['codice']."</td>";
   echo "<td>".$genera_clienti->clienti[$i]['ragione']."</td>";
   echo "<td>".$genera_clienti->clienti[$i]['id_categoria']."</td>";
   echo "<td>".$genera_clienti->clienti[$i]['ordini']."</td>";
   echo "<td>".$genera_clienti->clienti[$i]['articoli']."</td></tr>";
}
?>
</table>
```

Generare i fornitori

La generazione dei fornitori consiste nel riempire la tabella fornitori; occorre sapere solo il numero di fornitori.
Di seguito la classe relativa alla generazione dei fornitori

```
class genera_fornitori
{
   public $fornitori = array();
```

```
   protected $id_fornitore;
   protected $lunghezza_ragione;
   protected $genera_codice;

         public function __construct($lunghezza_ragione=20)
         {
            $this->id_fornitore=1;
            $this->lunghezza_ragione=$lunghezza_ragione;
            $this->genera_codice=new genera_codice('F');
   }

   public function genera($num)
   {
      for ($i=0; $i<$num; $i++)
      {
         $this->fornitori[$this->id_fornitore]['id']=$this->id_fornitore;
         $this->fornitori[$this->id_fornitore]['codice']=$this->genera_codice->successivo();
         $this->fornitori[$this->id_fornitore]['ragione']=get_frase($this->lunghezza_ragione);
         $this->id_fornitore++;
      }
   }
}
```

Di seguito un codice di esempio per generare i fornitori

```
<b>Generazione fornitori</b>
<table border="1" cellpadding="2" cellspacing="0">
  <tr><td><b>id</b></td>
  <td><b>codice</b></td>
  <td><b>ragione</b></td>
</tr>
<?php

$genera_fornitori = new genera_fornitori();
$genera_fornitori->genera(20);
for ($i=1; $i<=20; $i++)
{
   echo "<tr><td>".$genera_fornitori->fornitori[$i]['id']."</td>";
   echo "<td>".$genera_fornitori->fornitori[$i]['codice']."</td>";
   echo "<td>".$genera_fornitori->fornitori[$i]['ragione']."</td></tr>";
}
?>
</table>
```

Generare ordini clienti e ordini fornitori

Generare gli ordini dei clienti

La generazione degli ordini consiste nel riempire rispettivamente le tabelle ordine_cliente e ordine_cliente_dettaglio. Gli ordini vengono generati in ogni giorno lavorativo per un determinato lasso di tempo (minimo consigliato due anni), i clienti vengono scelti a caso in base alle assegnazioni dei volumi di vendita e per ogni ordine vengono selezionati gli articoli in modo da prendere con maggiore quantità gli articoli con alta scorta di sicurezza.
Di seguito la classe relativa alla generazione degli ordini

```
class genera_ordini_cliente
{
   public $ordini = array();
   public $ordini_dettaglio = array();
   protected $id_ordine;
   protected $id_ordine_dettaglio;
   protected $genera_clienti;
   protected $genera_articoli;
   protected $scegli_articolo;
   protected $scegli_qta;

   public function __construct(&$genera_clienti, &$genera_articoli, $scegli_articolo, $scegli_qta)
   {
      $this->id_ordine=1;
      $this->id_ordine_dettaglio=1;
      $this->genera_clienti=$genera_clienti;
      $this->genera_articoli=$genera_articoli;
      $this->scegli_articolo=$scegli_articolo;
      $this->scegli_qta=$scegli_qta;
   }

   public function genera($data)
   {
      $n_clienti=count($this->genera_clienti->clienti);
      for ($i=1; $i<=$n_clienti; $i++)
      {
         $volume_ordini=$this->genera_clienti->clienti[$i]['ordini'];
         if ($volume_ordini>0)
         {
            $is_ordine=rand(1,300);
            if ($is_ordine<$volume_ordini)
            {
               $this->genera_ordine($data, $i);
            }
         }
      }
   }

   protected function genera_ordine($data, $i)
   {
      $this->ordini[$this->id_ordine]['id']=$this->id_ordine;
```

```
      $this->ordini[$this->id_ordine]['data_ordine']=$data;
      $this->ordini[$this->id_ordine]['id_cliente']=$this->genera_clienti->clienti[$i]['id'];
      $this->ordini[$this->id_ordine]['nr_ordine']=$this->id_ordine;
      $funzione_scegli_articolo=$this->scegli_articolo;
      $funzione_scegli_qta=$this->scegli_qta;
      $n_righi=$this->genera_clienti->clienti[$i]['articoli'];
      $n_righi=get_random_media(1, $n_righi*2, 2);
      for ($j=1; $j<$n_righi; $j++)
      {
         $this->ordini_dettaglio[$this->id_ordine_dettaglio]['id']=$this->id_ordine_dettaglio;
         $this->ordini_dettaglio[$this->id_ordine_dettaglio]['id_ordine']=$this->id_ordine;
         $this->ordini_dettaglio[$this->id_ordine_dettaglio]['rigo']=$j;
         $id_articolo=$funzione_scegli_articolo();
         $this->ordini_dettaglio[$this->id_ordine_dettaglio]['id_articolo']=$this->genera_articoli-
>articoli[$id_articolo];
         $this->ordini_dettaglio[$this->id_ordine_dettaglio]['qta_ordinata']=$funzione_scegli_qta($id_articolo);
         $this->ordini_dettaglio[$this->id_ordine_dettaglio]['qta_evasa']=0;
         $this->id_ordine_dettaglio++;
      }
      $this->id_ordine++;
   }
}
```

La variabile $scegli_articolo contiene il riferimento alla funzione che determina l'articolo da inserire nel rigo d'ordine. La variabile scegli_qta contiene il riferimento alla funzione che determina la quantità da ordinare dell'articolo scelto. L'implementazione di queste due funzioni dipendono dal modo in cui si vuole che vengono scelti gli articoli e le quantità vendute. Di seguito un codice di esempio per generare gli ordini in una certa data.

```
function get_random_scorta_sicurezza()
{
   if (rand(1,3)<3) return 0;
   return get_random_decrescente(0, 1000, 2);
}

function get_random_volume_ordini()
{
   return get_random_media(0, 150, 2);
}

function get_random_volume_articoli()
{
   return get_random_decrescente(1, 10, 2);
}

function get_random_articolo()
{
   global $genera_articoli;
   $n_articoli=count($genera_articoli->articoli);
   while(true)
   {
      $articolo=rand(1, $n_articoli);
      if ($genera_articoli->articoli[$articolo]['scorta_sicurezza']>0) return $articolo;
      if (rand(1,5)==1) return $articolo;
   }
```

```
}


function get_random_qta($articolo)
{
   global $genera_articoli;
   $scorta_sicurezza=$genera_articoli->articoli[$articolo]['scorta_sicurezza'];
   if ($scorta_sicurezza<3) return 1;
   $qta=rand(1,$scorta_sicurezza/2);
   return $qta;
}

// generazione di 5 categorie
$genera_categorie = new genera_categorie();
$genera_categorie->genera(5);

// generazione di 100 articoli
$lista_caratteristiche = array(
array('AAS', 'BCB', 'XXX', 'ZZA'),
array('12', '14', '16', '18'),
array('05', '10', '20')
);

$genera_articoli = new genera_articoli('get_random_scorta_sicurezza');

for ($i=0; $i<5; $i++)
{
  $genera_codice = new genera_permutazione('A'.$i, $lista_caratteristiche);
  $genera_articoli->genera(20, $genera_codice, $i);
}

// generazione di 80 clienti
$genera_clienti = new genera_clienti('get_random_volume_ordini', 'get_random_volume_articoli');
$genera_clienti->genera(80, 5);

$genera_ordini_cliente=new genera_ordini_cliente($genera_clienti, $genera_articoli, 'get_random_articolo',
'get_random_qta');
$genera_ordini_cliente->genera('2008-09-12');

?>
<center>
<table cellpadding="5" cellspacing="5">
<tr>
<td><b>Generazione ordini</b>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
     <td><b>id</b></td>
     <td><b>data_ordine</b></td>
     <td><b>id_cliente</b></td>
     <td><b>nr_ordine</b></td>
  </tr>
<?php
$n_ordini=count($genera_ordini_cliente->ordini);
```

```
for ($i=1; $i<=$n_ordini; $i++)
{
   echo "<tr><td>".$genera_ordini_cliente->ordini[$i]['id']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini[$i]['data_ordine']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini[$i]['id_cliente']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini[$i]['nr_ordine']."</td></tr>";
}

?>
</table>
</td>
</tr>
<tr>
<td><b>Generazione dettaglio ordini</b>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
     <td><b>id</b></td>
     <td><b>id_ordine</b></td>
     <td><b>rigo</b></td>
     <td><b>id_articolo</b></td>
     <td><b>qta_ordinata</b></td>
     <td><b>qta_evasa</b></td>
  </tr>
<?php
$n_righi=count($genera_ordini_cliente->ordini_dettaglio);
for ($i=1; $i<=$n_righi; $i++)
{
   echo "<tr><td>".$genera_ordini_cliente->ordini_dettaglio[$i]['id']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini_dettaglio[$i]['id_ordine']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini_dettaglio[$i]['rigo']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini_dettaglio[$i]['id_articolo']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini_dettaglio[$i]['qta_ordinata']."</td>";
   echo "<td>".$genera_ordini_cliente->ordini_dettaglio[$i]['qta_evasa']."</td></tr>";
}

?>
</table>
</td>
</tr>
</table>
</center>
<?php
```

Si noti che la generazione degli ordini necessita di avere a disposizione, i dati degli articoli e dei clienti archiviati durante la loro generazione.

Inserire i dati nel database

Nei paragrafi precedenti abbiamo fratto vedere l'implementazione delle classi necessarie ad agevolare la creazione dei vari archivi, e abbiamo visto l'utilità di conservare i dati in array associativi. Adesso vedremo come inserire questi dati nel database. Di seguito la prima parte del codice relativa alla configurazione della generazione dei dati

```
// configurazione generazione

$numero_categorie=5;
```

```
$numero_clienti=100;
$numero_fornitori=20;
$numero_categorie_merceologiche=20;
$numero_articoli_categoria=100; // numero media di articoli per categoria

$inizio_anno=2008;
$periodo=10; // numero di giorni del processo di generazione ordini

$lista_caratteristiche = array(
array('ABC', 'BCB', 'GHS', 'XXX', 'ZZA'),
array('12', '14', '16', '18', '20', '24'),
array('05', '10', '20', '30', '40', '60'),
array('S', 'L', 'M', 'X')
);

function get_random_volume_ordini()
{
   return get_random_decrescente(0, 150, 3);
}

function get_random_volume_articoli()
{
   return get_random_decrescente(1, 10, 3);
}

function get_random_scorta_sicurezza()
{
   if (rand(1,3)<3) return 0;
   return get_random_decrescente(0, 1000, 2);
}

function get_random_articolo()
{
   global $genera_articoli;
   $n_articoli=count($genera_articoli->articoli);
   while(true)
   {
      $articolo=rand(1, $n_articoli);
      if ($genera_articoli->articoli[$articolo]['scorta_sicurezza']>0) return $articolo;
      if (rand(1,5)==1) return $articolo;
   }
}

function get_random_qta($articolo)
{
   global $genera_articoli;
   $scorta_sicurezza=$genera_articoli->articoli[$articolo]['scorta_sicurezza'];
   if ($scorta_sicurezza<3) return 1;
   $qta=rand(1,$scorta_sicurezza/2);
   return $qta;
}
```

Questa configurazione farà generare 100 clienti ed all'incirca 2000 articoli.

Il codice seguente serve per svuotare tutte le tabelle. Questa operazione consente di ripetere la generazione dei dati, utile nel caso in cui si vuole modificare la configurazione e ripetere l'operazione.

```
// reset delle tabelle

$sql = "TRUNCATE TABLE categoria ";
mysql_query($sql) or errore_db($sql);

$sql = "TRUNCATE TABLE cliente ";
mysql_query($sql) or errore_db($sql);

$sql = "TRUNCATE TABLE fornitore ";
mysql_query($sql) or errore_db($sql);

$sql = "TRUNCATE TABLE categoria_merceologica ";
mysql_query($sql) or errore_db($sql);

$sql = "TRUNCATE TABLE articolo ";
mysql_query($sql) or errore_db($sql);

$sql = "TRUNCATE TABLE ordine_cliente ";
mysql_query($sql) or errore_db($sql);

$sql = "TRUNCATE TABLE ordine_cliente_dettaglio ";
mysql_query($sql) or errore_db($sql);
```

Il codice seguente permette di inserire i dati anagrafici nel database

```
// creazione categorie
$genera_categorie = new genera_categorie();
$genera_categorie->genera($numero_categorie);

// inserimento delle categorie nel database
$sql=" INSERT INTO categoria (id, nome) VALUES ";
for ($i=1; $i<=$numero_categorie; $i++)
{
   if ($i>1) $sql .= ",";
   $id = $genera_categorie->categorie[$i]['id'];
   $nome = $genera_categorie->categorie[$i]['nome'];
   $sql .= "('$id', '$nome') ";
}
mysql_query($sql) or errore_db($sql);
echo "<br>Generate $numero_categorie categorie";



// creazione clienti
$genera_clienti = new genera_clienti('get_random_volume_ordini', 'get_random_volume_articoli');
$genera_clienti->genera($numero_clienti, $numero_categorie);

// inserimento dei clienti nel database
$sql=" INSERT INTO cliente (id, codice, ragione, id_categoria) VALUES ";
for ($i=1; $i<=$numero_clienti; $i++)
{
   if ($i>1) $sql .= ",";
   $id = $genera_clienti->clienti[$i]['id'];
```

```
    $codice = $genera_clienti->clienti[$i]['codice'];
    $ragione = $genera_clienti->clienti[$i]['ragione'];
    $id_categoria = $genera_clienti->clienti[$i]['id_categoria'];
    $sql .= "('$id', '$codice', '$ragione', '$id_categoria') ";
}
mysql_query($sql) or errore_db($sql);
echo "<br>Generati $numero_clienti clienti";


// creazione fornitori
$genera_fornitori = new genera_fornitori();
$genera_fornitori->genera($numero_fornitori);

// inserimento dei fornitori nel database
$sql=" INSERT INTO fornitore (id, codice, ragione) VALUES ";
for ($i=1; $i<=$numero_fornitori; $i++)
{
    if ($i>1) $sql .= ",";
    $id = $genera_fornitori->fornitori[$i]['id'];
    $codice = $genera_fornitori->fornitori[$i]['codice'];
    $ragione = $genera_fornitori->fornitori[$i]['ragione'];
    $sql .= "('$id', '$codice', '$ragione') ";
}
mysql_query($sql) or errore_db($sql);
echo "<br>Generati $numero_fornitori fornitori";

// creazione categorie merceologiche
$genera_categorie = new genera_categorie();
$genera_categorie->genera($numero_categorie_merceologiche);

// inserimento delle categorie nel database
$sql=" INSERT INTO categoria_merceologica (id, nome) VALUES ";
for ($i=1; $i<=$numero_categorie_merceologiche; $i++)
{
    if ($i>1) $sql .= ",";
    $id = $genera_categorie->categorie[$i]['id'];
    $nome = $genera_categorie->categorie[$i]['nome'];
    $sql .= "('$id', '$nome') ";
}
mysql_query($sql) or errore_db($sql);
echo "<br>Generate $numero_categorie_merceologiche categorie merceologiche";


// creazione articoli
$genera_articoli = new genera_articoli('get_random_scorta_sicurezza');
for ($i=1; $i<=$numero_categorie_merceologiche; $i++)
{
    $codice_cat=chr(rand(ord('A'), ord('Z'))).$i;
    $genera_codice = new genera_permutazione($codice_cat, $lista_caratteristiche);
    $articoli = rand($numero_articoli_categoria/2, $numero_articoli_categoria*2);
    $genera_articoli->genera($articoli, $genera_codice, $i);
}
// inserimento degli articoli nel database
```

```
$numero_articoli = count($genera_articoli->articoli);
$sql=" INSERT INTO articolo (id, codice, descrizione, scorta_fisica, scorta_sicurezza, id_categoria) VALUES
";
for ($i=1; $i<=$numero_articoli; $i++)
{
   if ($i>1) $sql .= ",";
   $id = $genera_articoli->articoli[$i]['id'];
   $codice = $genera_articoli->articoli[$i]['codice'];
   $descrizione = $genera_articoli->articoli[$i]['descrizione'];
   $id_categoria = $genera_articoli->articoli[$i]['id_categoria'];
   $scorta_sicurezza = $genera_articoli->articoli[$i]['scorta_sicurezza'];
   $scorta_fisica = $genera_articoli->articoli[$i]['scorta_fisica'];
   $sql .= " ('$id', '$codice', '$descrizione', '$scorta_fisica', '$scorta_sicurezza', '$id_categoria') ";
}

mysql_query($sql) or errore_db($sql);
echo "<br>Generati $numero_articoli articoli";
```

Infine il codice per inserire gli ordini

```
// creazione degli ordini
$numero_ordini=0;
$numero_ordini_dettaglio=0;
for ($j=1; $j<$periodo; $j++)
{
   $m=mktime(0, 0, 0, 1, $j, $inizio_anno);
   $settimana = date('w', $m);
   if (($settimana==0) || ($settimana==6) ) continue;
   $data=date('Y-m-d', $m);

   $genera_ordini_cliente=new genera_ordini_cliente($genera_clienti, $genera_articoli, 'get_random_articolo',
'get_random_qta');
   $genera_ordini_cliente->genera($data);
   $ordini = count($genera_ordini_cliente->ordini);
   $sql=" INSERT INTO ordine_cliente (id, data_ordine, id_cliente, nr_ordine) VALUES ";
   for ($i=1; $i<=$ordini; $i++)
   {
      if ($i>1) $sql .= ",";
      $id = $genera_ordini_cliente->ordini[$i]['id']+$numero_ordini;
      $data_ordine = $genera_ordini_cliente->ordini[$i]['data_ordine'];
      $id_cliente = $genera_ordini_cliente->ordini[$i]['id_cliente'];
      $nr_ordine = $genera_ordini_cliente->ordini[$i]['nr_ordine'];
      $sql .= " ('$id', '$data_ordine', '$id_cliente', '$nr_ordine') ";
   }
   $numero_ordini+=$ordini;
   mysql_query($sql) or errore_db($sql);

   $ordini_dettaglio = count($genera_ordini_cliente->ordini_dettaglio);
   $sql=" INSERT INTO ordine_cliente_dettaglio (id, id_ordine, rigo, id_articolo, qta_ordinata, qta_evasa, prezzo)
VALUES ";
   for ($i=1; $i<=$ordini_dettaglio; $i++)
   {
      if ($i>1) $sql .= ",";
      $id = $genera_ordini_cliente->ordini_dettaglio[$i]['id']+$numero_ordini_dettaglio;
```

```
      $id_ordine = $genera_ordini_cliente->ordini_dettaglio[$i]['id_ordine']+$numero_ordini;
      $rigo = $genera_ordini_cliente->ordini_dettaglio[$i]['rigo'];
      $id_articolo = $genera_ordini_cliente->ordini_dettaglio[$i]['id_articolo'];
      $qta_ordinata = $genera_ordini_cliente->ordini_dettaglio[$i]['qta_ordinata'];
      $qta_evasa = $genera_ordini_cliente->ordini_dettaglio[$i]['qta_evasa'];
      $prezzo = 0;
      $sql .= " ('$id', '$id_ordine', '$rigo', '$id_articolo', '$qta_ordinata', '$qta_evasa', '$prezzo') ";
   }
   $numero_ordini_dettaglio+=$ordini_dettaglio;
   mysql_query($sql) or errore_db($sql);
}

echo "<br>Generati $numero_ordini ordini con $numero_ordini_dettaglio righi ";
```

Generare gli ordini dei fornitori

La generazione degli ordini a fornitori in una certa data è subordinata sia da informazioni costanti come le scorta di sicurezza, sia da dati variabili come le giacenze, che dipendono dai movimenti effettuati fino al giorno prima. Altri dati variabili sono le quantità ancora da evadere degli ordini dei clienti e dei fornitori. Il primo passo per la generazione degli ordini a fornitore è il calcolo di quali articoli e in quali quantità occorre ordinare, il secondo passo occorre decidere da quale fornitore acquistare gli articoli, quindi infine generare gli ordini. Tralasciamo i primi due passi al paragrafo successivo e vediamo come generare gli ordini a partire da un vettore associativo contenente i dati degli ordini. Questi ultimi sono rispettivamente una lista di articoli, dove per ognuno di essi viene indicato il fornitore e la quantità da ordinare. Di seguito la classe relativa alla generazione degli ordini dei fornitori

```
class genera_ordini_fornitore
{
   public $ordini = array();
   public $ordini_dettaglio = array();
   protected $id_ordine;
   protected $id_ordine_dettaglio;

    public function __construct()
    {
       $this->id_ordine=1;
       $this->id_ordine_dettaglio=1;
   }

   public function genera($data, $lista=array())
   {
      $n_articoli=count($lista);
      $old_id_fornitore=0;
      for($i=0; $i<$n_articoli; $i++)
      {
         $id_fornitore = $lista[$i]['id_fornitore'];
         if ($old_id_fornitore!=$id_fornitore)
         {
            if ($old_id_fornitore>0) $this->id_ordine++;
            $this->ordini[$this->id_ordine]['id']=$this->id_ordine;
            $this->ordini[$this->id_ordine]['data_ordine']=$data;
            $this->ordini[$this->id_ordine]['id_fornitore']=$id_fornitore;
            $this->ordini[$this->id_ordine]['nr_ordine']=$this->id_ordine;
```

```
            $old_id_fornitore = $id_fornitore;
            $rigo=1;
        }
        $this->ordini_dettaglio[$this->id_ordine_dettaglio]['id']=$this->id_ordine_dettaglio;
        $this->ordini_dettaglio[$this->id_ordine_dettaglio]['id_ordine']=$this->id_ordine;
        $this->ordini_dettaglio[$this->id_ordine_dettaglio]['id_articolo']=$lista[$i]['id_articolo'];
        $this->ordini_dettaglio[$this->id_ordine_dettaglio]['qta_ordinata']=$lista[$i]['quantita'];
        $this->ordini_dettaglio[$this->id_ordine_dettaglio]['qta_evasa']=0;
        $this->ordini_dettaglio[$this->id_ordine_dettaglio]['rigo']=$rigo;
        $this->id_ordine_dettaglio++;
        $rigo++;
      }
   }

}
```

Si noti che la lista di articoli necessita un ordinamento dei fornitori
Di seguito un esempio per generare gli ordini dei fornitori in una certa data

```
function confronto_lista($i, $j)
{
   return ($i['id_fornitore']-$j['id_fornitore']);
}

$lista=array();
for($i=0; $i<50; $i++)
{
   $lista[$i]['id_fornitore']=rand(1,20);
   $lista[$i]['id_articolo']=rand(1,100);
   $lista[$i]['quantita']=rand(1,1000);
}

uasort($lista, 'confronto_lista');

$genera_ordini_fornitore=new genera_ordini_fornitore();
$genera_ordini_fornitore->genera('2008-09-12', $lista);

?>
<center>
<table cellpadding="5" cellspacing="5">
<tr>
<td><b>Generazione ordini dei fornitori</b>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
     <td><b>id</b></td>
     <td><b>data_ordine</b></td>
     <td><b>id_fornitore</b></td>
     <td><b>nr_ordine</b></td>
  </tr>
<?php
$n_ordini=count($genera_ordini_fornitore->ordini);
for ($i=1; $i<=$n_ordini; $i++)
{
   echo "<tr><td>".$genera_ordini_fornitore->ordini[$i]['id']."</td>";
```

```
    echo "<td>".$genera_ordini_fornitore->ordini[$i]['data_ordine']."</td>";
    echo "<td>".$genera_ordini_fornitore->ordini[$i]['id_fornitore']."</td>";
    echo "<td>".$genera_ordini_fornitore->ordini[$i]['nr_ordine']."</td></tr>";
}

?>
</table>
</td>
</tr>
<tr>
<td><b>Generazione dettaglio ordini dei fornitori</b>
<table border="1" cellpadding="2" cellspacing="0">
  <tr>
     <td><b>id</b></td>
     <td><b>id_ordine</b></td>
     <td><b>rigo</b></td>
     <td><b>id_articolo</b></td>
     <td><b>qta_ordinata</b></td>
     <td><b>qta_evasa</b></td>
  </tr>
<?php
$n_righi=count($genera_ordini_fornitore->ordini_dettaglio);
for ($i=1; $i<=$n_righi; $i++)
{
    echo "<tr><td>".$genera_ordini_fornitore->ordini_dettaglio[$i]['id']."</td>";
    echo "<td>".$genera_ordini_fornitore->ordini_dettaglio[$i]['id_ordine']."</td>";
    echo "<td>".$genera_ordini_fornitore->ordini_dettaglio[$i]['rigo']."</td>";
    echo "<td>".$genera_ordini_fornitore->ordini_dettaglio[$i]['id_articolo']."</td>";
    echo "<td>".$genera_ordini_fornitore->ordini_dettaglio[$i]['qta_ordinata']."</td>";
    echo "<td>".$genera_ordini_fornitore->ordini_dettaglio[$i]['qta_evasa']."</td></tr>";
}

?>
</table>
</td>
</tr>
</table>
</center>
<?php
```

Generare i movimenti di magazzino

Questo processo è stato inserito appositamente in ultimo poiché è quello più complesso. Le evasioni degli ordini sono determinati dagli ordini in archivio, e dalle giacenze in magazzino, e queste ultime sono determinate dalle evasioni degli ordini dei fornitori. In ultimo si tiene presente che gli ordini a fornitori sono state calcolate a loro volta dalle giacenze, e dalle scorta di sicurezza. Per rispecchiare fedelmente la realtà, occorre giorno per giorno, calcolare i nuovi ordini a fornitore, calcolare le giacenze di magazzino a seguito delle evasioni dei fornitori, determinare le evasioni degli ordini dei clienti, quindi rifare il calcolo delle giacenze. Per semplicità possiamo tralasciare i criteri principali di evasione degli ordini dei clienti (cioè il vincolo sul minimo importo evadibile e il vincolo sulla minima quantità evadibile), ottenendo un codice più snello e veloce. Sui criteri principali di evasione di ordini dei fornitori, occorre solamente predisporre un certo tempo di consegna, e per semplicità possiamo stabilire che i fornitori quando consegnano, evadono completamente l'ordine.

I sorgenti che vedremo in seguito possono essere inseriti all'interno del ciclo del calcolo degli ordini. Ovviamente occorre aggiungere queste impostazioni alla configurazione della generazione

```
// configurazione generazione
  ……
$consegna_fornitore=7;
function confronto_lista($i, $j)
{
   return ($i['id_fornitore']-$j['id_fornitore']);
}
  ……
```

Inoltre devono essere aggiunti il reset delle tabelle relative agli ordini a fornitore.

```
// reset delle tabelle
  ……
$sql = "TRUNCATE TABLE ordine_fornitore ";
mysql_query($sql) or errore_db($sql);

$sql = "TRUNCATE TABLE ordine_fornitore_dettaglio ";
mysql_query($sql) or errore_db($sql);
  ……
```

All'interno del ciclo, appena generati gli ordini dei clienti della data ($data), il primo passo è quello di calcolare gli articoli che possono essere consegnati ai clienti. Vediamo il codice SQL

```
for ($j=1; $j<$periodo; $j++)
{
   $m=mktime(0, 0, 0, 1, $j, $inizio_anno);
   $settimana = date('w', $m);
   if (($settimana==0) || ($settimana==6) ) continue;
   $data=date('Y-m-d', $m);


  ……

   $giacenze=array();
   $sql = "
     SELECT d.id_articolo, a.scorta_fisica
     FROM ordine_cliente_dettaglio AS d
     INNER JOIN articolo AS a ON a.id=d.id_articolo
     WHERE d.qta_ordinata>d.qta_evasa
     AND a.scorta_fisica>0
```

```
    GROUP BY d.id_articolo ";
  $res = mysql_query($sql) or errore_db($sql);
  while ($row = mysql_fetch_array($res))
  {
      $giacenze[$row['id_articolo']] = $row['scorta_fisica'];
  }
  ……
}
```

Si noti che l'array associativo contiene gli articoli con relativa giacenza, che hanno almeno un rigo d'ordine da evadere. Si osservi che il ciclo esclude automaticamente i sabati e le domeniche.

Ora è possibile assegnare le giacenze agli ordini dei clienti, considerando come priorità gli ordini con data più vecchia. Per ogni assegnazione, possiamo direttamente eseguire una evasione del rigo d'ordine del cliente ed una rettifica del magazzino. Le evasioni ovviamente avvengono fino a esaurimento scorte.

```
  ……

  $sql = " SELECT d.*
        FROM ordine_cliente AS o
        INNER JOIN ordine_cliente_dettaglio AS d ON o.id=d.id_ordine
        INNER JOIN articolo AS a ON a.id=d.id_articolo
        WHERE d.qta_ordinata>d.qta_evasa
        AND a.scorta_fisica>0
        ORDER BY o.data_ordine DESC ";
  $res = mysql_query($sql) or errore_db($sql);
  while ($row = mysql_fetch_array($res))
  {
    $id_articolo = $row['id_articolo'];
    $qta = $row['qta_ordinata']-$row['qta_evasa'];
    if ($giacenze[$id_articolo]>0)
    {
      if ($qta>$giacenze[$id_articolo]) $qta = $giacenze[$id_articolo];

      $sql = " UPDATE ordine_cliente_dettaglio
            SET qta_evasa = qta_evasa + '$qta'
            WHERE id = '$row[id]' ";
      mysql_query($sql) or errore_db($sql);

      $sql = " UPDATE articolo
            SET scorta_fisica = scorta_fisica - '$qta'
            WHERE id = '$id_articolo' ";
      mysql_query($sql) or errore_db($sql);

      $giacenze[$id_articolo] -= $qta;
    }
  }
  ……
```

Si noti che in questa fase, se il gestionale doveva emettere un DDT, essa avrebbe dovuto farlo all'interno di queste operazioni.

Il passo successivo è quello di determinare quali sono gli articoli da riordinare. Di seguito il codice SQL

```
  ……
  $sql_impegnato = "
    SELECT id_articolo, sum(qta_ordinata-qta_evasa) AS impegnato
```

```
    FROM ordine_cliente_dettaglio
    WHERE qta_ordinata>qta_evasa
    GROUP BY id_articolo ";

  $sql_ordinato = "
    SELECT id_articolo, sum(qta_ordinata-qta_evasa) AS ordinato
    FROM ordine_fornitore_dettaglio
    WHERE qta_ordinata>qta_evasa
    GROUP BY id_articolo ";

  $lista=array();
  $i=0;
  $sql = " SELECT a.id, a.scorta_fisica, a.scorta_sicurezza, c.impegnato, f.ordinato
          FROM articolo AS a
          LEFT JOIN ($sql_impegnato) AS c ON c.id_articolo=a.id
          LEFT JOIN ($sql_ordinato) AS f ON f.id_articolo=a.id
          WHERE (a.scorta_fisica
               + CASE WHEN f.ordinato IS NULL THEN 0 ELSE f.ordinato END)
               <
              (a.scorta_sicurezza
               + CASE WHEN c.impegnato IS NULL THEN 0 ELSE c.impegnato END) ";
  $res = mysql_query($sql) or errore_db($sql);
  while ($row = mysql_fetch_array($res))
  {
     $id_articolo = $row['id'];
     $lista[$i]['id_articolo'] = $id_articolo;
     $scorta_fisica = $row['scorta_fisica'];
     $scorta_sicurezza = $row['scorta_sicurezza'];
     $impegnato = $row['impegnato'];
     if (!$impegnato) $impegnato=0;
     $ordinato = $row['ordinato'];
     if (!$ordinato) $ordinato=0;
     if (!$articoli_fornitori[$id_articolo])
     {
        $articoli_fornitori[$id_articolo] = rand(1,$numero_fornitori);
     }
     $lista[$i]['id_fornitore'] = $articoli_fornitori[$id_articolo];
     $lista[$i]['quantita'] = $scorta_sicurezza+$impegnato-($scorta_fisica+$ordinato);
     $i++;
  }
  ……
```

Si noti il vettore associativo $articoli_fornitori utilizzato per associare gli articoli ai fornitori, in pratica per ogni articolo viene scelto un unico fornitore.

A questo punto è abbastanza semplice calcolare gli ordini a fornitore

```
  ……
  uasort($lista, 'confronto_lista');

  $genera_ordini_fornitore=new genera_ordini_fornitore();
  $genera_ordini_fornitore->genera($data, $lista);

  $ordini = count($genera_ordini_fornitore->ordini);
  $sql=" INSERT INTO ordine_fornitore (id, data_ordine, id_fornitore, nr_ordine) VALUES ";
```

```
   for ($i=1; $i<=$ordini; $i++)
   {
      if ($i>1) $sql .= ",";
      $id = $genera_ordini_fornitore->ordini[$i]['id']+$numero_ordini_fornitore;
      $data_ordine = $genera_ordini_fornitore->ordini[$i]['data_ordine'];
      $id_fornitore = $genera_ordini_fornitore->ordini[$i]['id_fornitore'];
      $nr_ordine = $genera_ordini_fornitore->ordini[$i]['nr_ordine'];
      $sql .= " ('$id', '$data_ordine', '$id_fornitore', '$nr_ordine') ";
   }
   mysql_query($sql) or errore_db($sql);

   $ordini_dettaglio = count($genera_ordini_fornitore->ordini_dettaglio);
   $sql=" INSERT INTO ordine_fornitore_dettaglio (id, id_ordine, rigo, id_articolo, qta_ordinata, qta_evasa,
prezzo) VALUES ";
   for ($i=1; $i<=$ordini_dettaglio; $i++)
   {
      if ($i>1) $sql .= ",";
      $id = $genera_ordini_fornitore->ordini_dettaglio[$i]['id']+$numero_ordini_fornitore_dettaglio;
      $id_ordine = $genera_ordini_fornitore->ordini_dettaglio[$i]['id_ordine']+$numero_ordini_fornitore;
      $rigo = $genera_ordini_fornitore->ordini_dettaglio[$i]['rigo'];
      $id_articolo = $genera_ordini_fornitore->ordini_dettaglio[$i]['id_articolo'];
      $qta_ordinata = $genera_ordini_fornitore->ordini_dettaglio[$i]['qta_ordinata'];
      $qta_evasa = $genera_ordini_fornitore->ordini_dettaglio[$i]['qta_evasa'];
      $prezzo = 0;
      $sql .= " ('$id', '$id_ordine', '$rigo', '$id_articolo', '$qta_ordinata', '$qta_evasa', '$prezzo') ";
   }
   $numero_ordini_fornitore_dettaglio+=$ordini_dettaglio;
   $numero_ordini_fornitore+=$ordini;
   mysql_query($sql) or errore_db($sql);
   ……
```

A questo punto si può procedere con l'evasione degli ordini a fornitore e al ripristino delle giacenze. Si noti l'uso della variabile $consegna_fornitore per calcolare l'effettiva data di consegna del fornitore.

```
   ……
   $m=mktime(0, 0, 0, 1, $j-$consegna_fornitore, $inizio_anno);
   $data=date('Y-m-d', $m);

   $sql = " SELECT d.*
         FROM ordine_fornitore AS o
         INNER JOIN ordine_fornitore_dettaglio AS d ON o.id=d.id_ordine
         WHERE o.data_ordine<='$data'
         AND d.qta_evasa<d.qta_ordinata ";
   $res = mysql_query($sql) or errore_db($sql);
   while ($row = mysql_fetch_array($res))
   {
      $id_articolo = $row['id_articolo'];
      $qta = $row['qta_ordinata']-$row['qta_evasa'];

      $sql = " UPDATE ordine_fornitore_dettaglio
            SET qta_evasa = qta_ordinata
            WHERE id = '$row[id]' ";
      mysql_query($sql) or errore_db($sql);
```

```
    $sql = " UPDATE articolo
            SET scorta_fisica = scorta_fisica + '$qta'
            WHERE id = '$id_articolo' ";
    mysql_query($sql) or errore_db($sql);

}
……
```

Coclusioni

Se questo documento ti è piaciuto e vorresti contribuire alla sua sua diffusione, puoi farlo. Ecco alcuni suggerimenti:

- **Passaparola**: è il metodo più semplice per consigliare questo documento a chi lo sta cercando;
- **Segnalazioni**: potete segnalarlo a siti che offrono risorse gratuite;
- **Scambio dati**: potete inserire il PDF nei vostri archivi di programmi di file sharing;
- **Link o banner**: potete inserire sul vostro sito un link sul sito o direttamente sul documento PDF;

Se trovi un errore di battitura, o una inesattezza, o qualunque altra cosa "strana" e stai leggendo l'ultima versione, puoi segnalare il problema alla mia e-mail **info@francescofiora.it**.
In oltre suggerimenti e osservazioni sono i benvenuti, e se vuoi puoi anche segnalarmi quanche altro argomento che vorresti che ne fosse discusso nelle successive versioni del documento.