

Interazione¹ linguaggio PHP con DB *MySQL*

Un problema sempre più frequente dei webmaster è quello di dover gestire intere sezioni dei loro siti in modo semplice e veloce. La gestione è quanto più efficiente, tanto più è possibile effettuare **modifiche frequenti** di contenuto, ma, a volte, anche di veste grafica. Un sito che aspiri ad aver un certo successo deve anche offrire una **consistente quantità di informazioni**, ma è impensabile dover modificare centinaia di pagine ogni volta si apportino anche il minimo aggiornamento.

In tutto questo, per fortuna, ci sono venuti in aiuto i linguaggi **orientati al web publishing** come PHP, ASP o Perl. Purtroppo tutto questo si è rivelato subito insufficiente: i linguaggi sono ottimi per la creazione di **pagine dinamiche**, ma non offrono nessuna possibilità di memorizzazione dei dati. La soluzione attuale è quindi quella di utilizzare **parallelamente** tali **linguaggi** ed un **database**.

Ad esempio si fa interagire **PHP** con il database relazionale **MySQL**. Questa accoppiata è, al giorno d'oggi, una delle più diffuse in rete in quanto abbiamo a disposizione gratuitamente un linguaggio solido, capace di sopportare grandi carichi di lavoro, e un database (in realtà un DBMS) dalle notevoli qualità tecniche.

È importante prima di tutto chiarire un concetto: si ha a disposizione un database che non viene memorizzato in un file specifico. In particolare questo viene memorizzato in un **insieme di file** che non è accessibile a chiunque. Anche se possedete un dominio e vi siete rivolti a un servizio di hosting a pagamento, questo non vi permetterà di accedere ai file: potrete modificarli indirettamente (quindi tramite query), ma non potrete copiarli o salvarli. Questo crea una limitazione nel senso che quando volete distribuire uno script, non potete fornire con esso anche il database. Dovete quindi fare in modo che l'utente crei le tabelle necessarie all'interno del suo database: potremmo quindi fornire all'utente accesso ad interfacce (come **phpMyAdmin**²) per crearsi tabelle con determinate caratteristiche (sfruttando funzionalità di *esportazione/importazione*) ma questo metodo richiede una serie di conoscenze da parte dell'utente che probabilmente non ha. Quindi gli forniremo uno script che creerà per lui tutte le tabelle necessarie.

Connessione al database

Innanzitutto vedremo le operazioni necessarie per la **connessione al database**.

Prima di poter comunicare con un DB abbiamo infatti bisogno di creare un "collegamento" fra lo script e MySQL.

Avremo bisogno di alcune informazioni relative all'accesso al database: l'**host** da cui si può raggiungere MySQL (generalmente è *localhost*); **username** e **password** per l'accesso al database; il **nome** del database. Questi quattro parametri vengono forniti dall'amministratore del nostro spazio web e prevediamo di memorizzare in altrettante variabili tali dati relativi alla configurazione dello script che salveremo con estensione **.php** per motivi di sicurezza.

Se qualcuno cercasse, infatti, di visualizzare questa pagina con il browser, vedrebbe solo una pagina vuota.

Il webserver, infatti, grazie a quest'estensione, prima di passare la pagina al browser, la farà elaborare dal modulo Php. Visto che non è previsto nessun output, sul browser verrà visualizzata solo una pagina bianca.

¹ <http://www.html.it/guide/guida-php-e-mysql-pratica/>

² **PhpMyAdmin** non è altro che un'interfaccia grafica che permette di amministrare **MySQL**, un tipo di database che immagazzina qualsiasi tipo di dati in strutture chiamate tabelle; con PhpMyAdmin, in pratica, si può visualizzare il contenuto di un database; creare, modificare, cancellare intere tabelle o singoli record; fare un backup dei dati contenuti; visualizzare informazioni interessanti sul db.

```

// parametri di connessione
$db_host = "localhost";           // il server che si trova sulla macchina locale
$db_user = "";
$db_password = "";

// parametri del database
$db_name = "";

```

Per **connettersi**, allora, ad un database *MySQL* remoto, usando l'*estensione nativa*³ a *MySQL*, useremo la funzione **mysql_connect()** che prevede come argomenti l'**host** da cui si può raggiungere MySQL, **username** e **password** per l'accesso al database:

mysql_connect (server, login, password);

Tale funzione consente il collegamento con il database. I valori da inserire solitamente sono dati dal server dove risiede il database (in fase di registrazione utente dovete inserire la password ed il login). Questa funzione se riceve un valore lo passa alla variabile *resource*⁴ (nel nostro esempio `$db`).

```

<?
$db = mysql_connect($db_host, $db_user, $db_password);
?>

```

Se la connessione ha buon fine restituisce un *identificatore* alla connessione che si memorizza nella variabile `$db`. Useremo questa variabile ogni volta che vorremo fare un'operazione sul database.

Se la connessione non dovesse andare a buon fine (per esempio se uno dei parametri fosse sbagliato) verrebbe restituito FALSE.

Quindi dovremo verificare il buon esito della connessione utilizzando la funzione **die()** che interrompe l'esecuzione dopo aver scritto a video tutto quello che contengono le parentesi:

```

<?
if ($db == FALSE)
die ("Errore nella connessione5. Verificare i parametri");
?>

```

Una volta stabilita la connessione, il passo successivo è **selezionare il database** col quale vogliamo lavorare. Per questo si usa la funzione **mysql_select_db(nomedb,connessione)**: col primo parametro passiamo il nome del db al quale vogliamo connetterci, col secondo l'identificativo di connessione (cioè quello che abbiamo ottenuto da `mysql_connect`). Questa funzione **restituisce** un valore booleano che indica se la selezione del database è riuscita o no.

Quindi si prevede di verificare con quale database si vuole lavorare e di controllare nuovamente la riuscita dell'operazione:

```

<?
mysql_select_db($db_name, $db)
or die ("Errore nella selezione del database. Verificare i parametri");
?>

```

³ Dalla versione PHP 2.0; dalla versione [PHP 7](#) tale interfaccia non è più supportata, sostituita da altre due interfacce: l'*estensione Mysqli* da PHP 5.0 e l'*estensione* PHP Data Objects (*PDO*) da PHP 5.1

⁴ Nuovo tipo di variabile che ci serve come *puntatore* al database

⁵ Volendo conoscere il motivo della mancata connessione, si usa la funzione senza parametri **mysql_error()** che effettua la stampa dell'errore segnalato dal server MySQL impostando la seguente sintassi:

die("Errore nella connessione a MySQL: " . mysql_error());

Esecuzione di una query

Si è arrivati alla parte fondamentale del colloquio con un database, cioè l'esecuzione di una query. Per eseguire la query si usa la funzione `mysql_query(query,connessione)`, alla quale viene passata la query da eseguire insieme all'identificativo di connessione. Anche questa funzione **restituisce** un valore, per il quale però dobbiamo distinguere due possibilità rispetto al tipo di query che abbiamo lanciato:

- Se si tratta di una **query di interrogazione** (SELECT, SHOW, EXPLAIN, DESCRIBE), la funzione restituisce un **identificativo del risultato** (cioè un'altra variabile di tipo resource), che ci servirà successivamente, se la query è andata a buon fine; se invece MySQL ha rilevato degli errori, la funzione restituisce FALSE;
- Se invece si tratta di una **query di aggiornamento** (INSERT, UPDATE, DELETE e tutte le altre diverse da quelle viste prima), la funzione restituirà in ogni caso un valore booleano, ad indicare se l'esecuzione è andata a buon fine oppure no.

A questo punto dobbiamo richiamare l'attenzione ancora una volta sulla necessità di **verificare il risultato** della nostra query, importante più che mai in questa situazione in quanto è molto facile commettere errori in una query. Vediamo quindi un esempio:

```
$query = 'SELECT * FROM tabella';  
$ris = mysql_query($query,$db) or die("Errore nella query: " . mysql_error());
```

In questo modo, se la query ha avuto successo la variabile \$ris conterrà l'identificativo del risultato, che ci servirà successivamente per **leggere** le righe restituite dal db. Se invece la query non va a buon fine, lo script si blocca segnalando l'errore. Nel caso in cui avessimo voluto eseguire una query di aggiornamento, avremmo potuto evitare di assegnare il risultato ad una variabile.

Verifica dei risultati della query

Il fatto che una query sia stata eseguita correttamente non significa necessariamente che abbia prodotto dei risultati. Può infatti verificarsi il caso in cui una query, pur essendo perfettamente corretta, non produce alcun risultato, ad esempio perché le condizioni che abbiamo specificato nella clausola WHERE non sono mai verificate sulle tabelle interessate.

Se vogliamo sapere **quante** righe sono state **restituite** da una SELECT, possiamo usare la funzione `mysql_num_rows(risultato)`, che ci **restituisce** il numero di righe contenute dall'identificativo del risultato che le passiamo.

Se invece abbiamo eseguito una query di aggiornamento (INSERT, UPDATE, DELETE) e vogliamo sapere **quante** righe sono state **modificate**, possiamo usare `mysql_affected_rows(connessione)`, che ci **restituisce** il numero di righe modificate dall'ultima query di aggiornamento.

```
$query = 'SELECT * FROM tabella';  
$ris = mysql_query($query,$db) or die("Errore nella query: " . mysql_error());  
  
$righe = mysql_num_rows($ris);          /* $righe riceve il numero di righe restituite dalla SELECT */  
  
$query = "UPDATE tabella SET campo1='valore' WHERE campo2='val'";  
mysql_query($query,$db) or die("Errore nella query: " . mysql_error());  
  
$righe = mysql_affected_rows($db);     /* $righe riceve il numero di righe modificate da UPDATE */
```

È importante notare la differenza nel parametro da passare alle due funzioni: mentre `mysql_num_rows()` richiede un *identificativo di risultato*, `mysql_affected_rows()` richiede un *identificativo di connessione*; infatti, come abbiamo visto prima, una query di aggiornamento **non** restituisce un identificativo di risultato.

Letture dei risultati di una SELECT

Come abbiamo visto prima, una volta effettuata una query di interrogazione abbiamo a disposizione un identificativo del suo risultato. Per poter leggere questo risultato possiamo utilizzare la funzione **mysql_fetch_array(risultato)**, la quale, **ogni volta** che viene chiamata, ci **restituisce una riga** del nostro risultato; quando non ci sono più righe da leggere, la funzione **restituisce FALSE**. Quindi, per scorrere tutto il risultato, dovremo usare questa funzione come condizione di un ciclo, che si concluderà quando restituisce FALSE. In questo modo **non abbiamo bisogno** di sapere a priori quante sono le righe contenute nel risultato stesso.

```
$query = 'SELECT * FROM tabella';

$ris = mysql_query($query,$db) or die("Errore nella query: " . mysql_error());

while($riga = mysql_fetch_array($ris))
{
    //codice che elabora i dati
}
```

Ogni volta che questo ciclo viene eseguito, quindi, avremo a disposizione, nella variabile \$riga, una riga del nostro risultato. Questa variabile è in effetti un **array** che contiene i valori delle colonne restituiti dalla nostra query. Gli indici dell'array sono i nomi delle colonne, ed i loro valori sono i valori estratti dal database. Rivediamo dunque l'esempio di prima, specificando per maggior chiarezza quali colonne vogliamo estrarre dalla tabella:

```
$query = 'SELECT nome, indirizzo, telefono FROM tabella';
$ris = mysql_query($query,$db) or die("Errore nella query: " . mysql_error());
while($riga = mysql_fetch_array($ris)) {
    print"Nome: $riga[nome]<br/>";
    print"Indirizzo: $riga[indirizzo]<br/>";
    print"Telefono: $riga[telefono]<br/>";
    print"<br/>";
}
```

Con questo ciclo quindi stamperemo tutti i valori estratti dalla query, separando con una riga vuota i blocchi relativi ad ogni record. Nella select abbiamo estratto le colonne 'nome', 'indirizzo' e 'telefono', e quindi l'array \$riga conterrà tre elementi con questi indici. In realtà, l'array \$riga contiene anche altri tre elementi, con indici numerici 0, 1 e 2, che contengono sempre gli stessi dati (nome, indirizzo e telefono) nell'ordine in cui li abbiamo indicati nella select.

Questi dati sono alquanto inutili, in quanto è molto più comodo, ovviamente, usare gli indici alfanumerici con i nomi delle colonne. Se vogliamo evitare di ricevere questi dati aggiuntivi, togliendo così un po' di lavoro a PHP, possiamo chiamare la funzione **mysql_fetch_array()** specificando il parametro aggiuntivo **MYSQL_ASSOC** (è una costante, va scritta in maiuscolo senza \$ davanti); in alternativa, possiamo usare la funzione **mysql_fetch_assoc(risultato)**, che equivale a **mysql_fetch_array()** ma restituisce solo gli indici associativi.

```
$riga = mysql_fetch_array($ris,MYSQL_ASSOC);           /* solo indici associativi */
$riga = mysql_fetch_assoc($ris);                       // solo indici associativi
```

Con l'introduzione⁶ del **modello ad oggetti**, possiamo recuperare come oggetti le t-ple che costituiscono il *recordset* cioè la tabella dinamica generata da una query di interrogazione; si prelevano infatti le singole righe della tabella, usando la funzione:

mysql_fetch_object (risultato)

che restituisce la riga corrente, potendo recuperare i valori di ogni campo come **attributi di oggetti**:

```
$riga = mysql_fetch_object($ris);          /* preleva la riga corrente della tabella dinamica */

while($riga = mysql_fetch_object($ris))
{
    echo "$riga->nome &nbsp; $riga->telefono &nbsp; $riga->indirizzo <br>\n";
    // visualizza scorrendo tutte le righe ed inserisce uno spazio tra valori in ogni riga
}
```

oppure, sfruttando la possibilità di gestire con echo più stringhe divise da virgole:

```
$riga = mysql_fetch_object($ris); /* preleva la riga corrente della tabella dinamica */

while($riga = mysql_fetch_object($ris))
{
    echo "$riga->nome" , " &nbsp; $riga->telefono" , " &nbsp; $riga->indirizzo <br>\n" ;
    // visualizza scorrendo tutte le righe ed inserisce uno spazio tra valori in ogni riga
}
```

Si è quindi completata questa veloce carrellata sulle principali **funzioni** (l'*estensione nativa a MySQL*) da utilizzare per interagire da PHP con un database MySQL.

Rimane da citare la funzione **mysql_close (connessione)**, che serve per chiudere la connessione aperta con **mysql_connect()**, ma in pratica questa funzione è usata pochissimo, in quanto PHP si preoccupa da solo, al termine dello script, di chiudere le connessioni che abbiamo aperto.

```
mysql_close($db);
```

```
mysql_close(); // non specificando, saranno chiuse tutte le connessioni
```

nb: in PHP 7 non sono più incluse le **mysql functions** che non vanno più considerate un'alternativa valida e si usa l'*estensione MySQLi (MySQL improved)*: una caratteristica peculiare di questa estensione è quella di disporre di una doppia interfaccia: **procedurale** e **ad oggetti**.

Una terza possibilità è l'uso dell'*estensione PHP Data Objects (PDO)* completamente orientata agli oggetti, e che non può quindi essere utilizzata con un approccio procedurale. La caratteristica principale che differenzia questa estensione rispetto a MySQLi è che essa dispone di driver per **diversi DBMS**, non limitandosi esclusivamente a MySQL. Questo significa che se anche la nostra applicazione è stata progettata per comunicare con MySQL possiamo adattarla, con piccole variazioni del codice, ad altri software. Tra i *driver* disponibili, quelli per i seguenti DBMS: Cubrid, Microsoft SQL Server, Firebird, IBM DB2, Informix Dynamic Server, MySQL, Oracle, ODBC e DB2, PostgreSQL, **SQLite**, 4D.

In sostanza **PDO** (nel confronto) fornisce un *livello di astrazione* per l'interazione con le basi di dati.

Esempio di connessione a MySQL con *estensione mysql* nell'**approccio ad oggetti**:

```
$mysqli = new mysqli('localhost', 'username', 'password', 'nome_database');
if ($mysqli->connect_error) {
    die('Errore di connessione (' . $mysqli->connect_errno . ') '
        . $mysqli->connect_error);
} else {
    echo 'Connesso. ' . $mysqli->host_info . "\n"; }
}
```

⁶ Supporto di base agli oggetti da **PHP 4.0**; supporto completo agli oggetti e migliori performance da **PHP 5.0**

Creazione di una tabella

Per creare una tabella e controllare la riuscita dell'operazione:

<?

```
$query = "CREATE TABLE news (id INT (5) UNSIGNED not null AUTO_INCREMENT, titolo VARCHAR (255) not null , testo TEXT not null , data INT (11) , autore VARCHAR (50) , mail VARCHAR (50) , PRIMARY KEY (id));"
```

```
// controllo usando l'estensione nativa a MySQL
```

```
if (mysql_query($query, $db))
```

```
    echo "L'installazione e' stata eseguita correttamente";
```

```
else
```

```
    echo "Errore durante l'installazione";
```

```
mysql_close($db);
```

```
?>
```



Inserimento di record in una tabella

Esempio di inserimento con campi obbligatori titolo e testo poichè id si autoincrementa

Nb: se non forniamo il contenuto dei campi not null, l'inserzione non può avvenire.

<?

```
$query = "INSERT INTO news (titolo, testo, data, autore, mail) VALUES ('vecchio articolo', 'Ecco un articolo', '1002664800', 'prof. Biasotti', 'pbiasotti@libero.it');"
```

```
// esecuzione e controllo usando l'estensione nativa a MySQL
```

```
// con data 1002664800 in formato timestamp
```

```
// cioè numero di secondi trascorsi a partire dall'ora 00:00 del 1 gennaio 1970
```

```
if (mysql_query($query, $db))
```

```
    echo "L'articolo e' stato inserito correttamente";
```

```
else
```

```
    echo "Errore durante l'inserimento";
```

```
echo "<br/>";
```

```
?>
```



NB: analogamente, si potranno realizzare aggiornamenti

o cancellazioni



Estrazione da una tabella

Esempio di estrazione e visualizzazione con uso di identificatore delle singole righe cioè la variabile \$result usando l'*estensione nativa a MySQL*

```
<?
$query = "SELECT * FROM news ";

$result = mysql_query($query, $db);

while ($row = mysql_fetch_array($result))

{ echo "$row[testo] " . date("j/n/y", $row[data]) . " - $row[titolo] <br/>"; }

/* date("j/n/y", $row[data]) è una funzione PHP
   che in base a una data in timestamp (nel nostro caso memorizzato in $row[data]
   crea la data nel formato g/m/aa.
*/

// funzione opzionale infatti la chiusura della connessione al database è automatica
// tutte le connessioni vengono chiuse automaticamente alla fine della pagina.

mysql_close($db);
?>
```



Esempio di estrazione in **ordine cronologico** di **sole 3 righe**:

```
<?
$query = "SELECT id,data,titolo FROM news ORDER BY data DESC LIMIT 0,3";
?>
```



NB: analogamente, si potranno realizzare estrazioni da più tabelle (*join*)

con vari criteri di selezione



e ancora si potranno deallocazione tabelle, indici, database

