

PHP e MySQL

E' possibile realizzare delle applicazioni in php appoggiandosi ad un database, quale ad esempio MySQL. Con le novità introdotte ai tempi di MySQL 4.1 il team di PHP ha sviluppato una nuova API (insieme di funzioni) che permette a PHP di interagire con MySQL, contenuta in una estensione chiamata **mysqli** ([mysqli improved](#)). Tale estensione si affianca quindi alla più vecchia estensione **mysql**, estendendone le funzionalità ed introducendo anche un'interessante interfaccia ad oggetti. Se si vogliono provare le nuove caratteristiche di MySQL 5 dovrete installare l'estensione **mysqli** mentre se vorrete testare la compatibilità di script PHP di non recente realizzazione l'estensione **mysql** farà al caso vostro. E' comunque possibile installarle entrambe.

Vi rimando comunque alla [guida di MySQL](#) per ulteriori approfondimenti sui database e alla [guida di Apache](#) per la sua configurazione per MySQL 5.

Nel PHP ci sono quindi delle funzioni native riservate proprio alla gestione di applicazioni su database. Di seguito sono mostrate le funzioni relative alla vecchia estensione di MySQL "mysql_" visto che su molti server web è ancora installata la versione del PHP 4.x; potete applicare le stesse funzioni anche alla nuova estensione di PHP5, usando però il suffisso "mysqli_".

mysql_connect(server,nome_utente,password)
mysql_connect("localhost:3306","root","")

Stabilisce la connessione al server MySQL.

mysql_close(identificativo_connessione);

Chiude la connessione al server MySQL.

mysql_create_db("nome_db");

Crea un database di nome "nome_db".

die();

Funziona come la funzione *exit()*; , solo che mi consente di fornire una frase in output scritta all'interno delle parentesi.

mysql_error(identificativo_connessione);

Cattura gli eventuali errori di MySQL e me li fornisce in output. L'identificativo di connessione può essere anche omissso.

mysql_select_db("nome_db",identificativo_connessione);

Corrisponde al comando *use* di MySQL.

mysql_query("stringa_query",identificativo_connessione);

Invia una query al database.

**mysql_db_query("nome_db",
"stringa_query",identificativo_connessione);**

Seleziona il database ed eseguisce la query inviata.

| | |
|---|---|
| <code>mysql_list_dbs(identificativo_connessione);</code> | Elenca tutti i database presenti. |
| <code>mysql_list_tables("nome_db",identificativo_connessione);</code> | Elenca tutte le tabelle del database passato come parametro. |
| <code>mysql_fetch_array("nome_array",MYSQL_ASSOC);</code> | Riceve come primo parametro un array contenente la query da passare a MySQL. Oltre a memorizzare i dati del risultato in un array con indice numerico, questa li memorizza anche con indici associativi usando i nomi dei campi come chiavi. Vedi l'esempio successivo per capire meglio tale associazione. |
| <code>mysql_fetch_assoc("nome_array");</code> | E' la stessa della precedente. |
| <code>mysql_free_result(\$risultato);</code> | Libera la memoria occupata dal risultato. Restituisce <i>TRUE</i> in caso di successo, <i>FALSE</i> in caso di fallimento. |
| <code>mysql_num_fields(\$risultato);</code> | Mi restituisce il numero di campi totali della tabella. |
| <code>mysql_num_rows(\$risultato);</code> | Mi restituisce il numero dei records totali. |
| <code>mysql_field_table(\$risultato,identificativo_campo);</code> | Mi restituisce il nome della tabella contenente l'identificativo di campo passato come secondo parametro. |
| <code>mysql_field_type(\$risultato,\$i);</code> | Mi restituisce il tipo di campo, quali "int", "real", "string", "bool". I due parametri sono rispettivamente la query e l'identificativo della cella. |
| <code>mysql_field_name(\$risultato,\$i);</code> | Mi restituisce il nome del campo. |

`mysql_field_len($risultato,$i);`

Mi restituisce la lunghezza dei caratteri assegnata al campo.

PHP e MySQL: ESEMPI

Vediamo ora qualche semplice esempio di applicazione delle funzioni viste nel paragrafo precedente; in primo luogo vediamo come realizzare una applicazione php che si colleghi al database, printi l'identificativo della connessione (esso è un numero che identifica la connessione esistente tra il vostro client ed il server di mysql) e poi che la chiudi.

```
<?php
$connessione = mysql_connect("localhost","root","") or die("Connessione non riuscita: ".mysql_error());
print("Identificativo di Connessione : ".$connessione);
mysql_close($connessione);
?>
```

Di risultato: *Identificativo di Connessione : Resource id #14*

Di seguito è mostrato un altro esempio: supponendo di avere un database *rubrica*, con una tabella *nomi* contenente appunto solo i nomi, si vuole printare l'elenco di tutti i nomi presenti nel database.

```
<?php
$connessione = mysql_connect("localhost","root","");
mysql_select_db("rubrica") or die("Selezione del database non riuscita");
$risultato = mysql_query("SELECT * FROM nomi",$connessione);
while($riga=mysql_fetch_array($risultato,MYSQL_ASSOC))
{
    echo("ID: ".$riga['id']." NOME: ".$riga['nome']."<br>");
}
mysql_free_result($risultato);
mysql_close($connessione);
?>
```

Ad esempio il risultato potrebbe essere:

```
ID: 1 Nome: Andrea
ID: 2 Nome: Sara
ID: 3 Nome: Marco
ID: 4 Nome: Salvo
```

Le chiavi passate nell'array \$riga devono essere scritte esattamente (maiuscolo/minuscolo) come sono state scritte in MySql, altrimenti la query non funziona. Vediamo ora in dettaglio la funzione `mysql_fetch_array`, a cui viene passato come parametro `MYSQL_ASSOC`: essa mi ricostruisce in maniera ricorsiva l'array \$riga associato al risultato della query mysql nel seguente modo:

```
Array(
    ['id']=>1,
    ['nome']=>"Andrea"
);
```

Alla seconda iterazione del ciclo while, ovviamente i valori cambierebbero.

Vediamo ora come printare i risultati di una query in una tabella realizzata con il semplice codice HTML, nel caso in cui si ha a che fare con una tabella *studenti* di un database chiamato *università*:

```
<?php
$connessione=mysql_connect("localhost","root","");
mysql_select_db("uni") or die("Selezione del database non riuscita");
$query="SELECT * FROM studenti";
$resultato = mysql_query($query) or die("Query non valida: ".mysql_error());
print("<table border=1 height=50% width=50% bgcolor=#66CCFF>\n");
while($riga=mysql_fetch_array($resultato,MYSQL_ASSOC))
{
    print("\t<TR>\n");
    foreach($riga as $key=>$val)
    {
        print("\t\t<TD>$val</TD>\n");
    }
    print("\t</TR>\n");
}
print("</TABLE>\n");
mysql_close($connessione);
?>
```

\t serve per inserire uno spazio di tabulazione, mentre \n per andare a capo; queste tabulazioni saranno visibili solo nel codice HTML della pagina visualizzata nel browser (*Visualizza -> HTML* dal menù del browser), e non servono per la formattazione della tabella di output.

Il cui risultato è il seguente:

| | | |
|---|--------|-----------|
| 1 | Andrea | De Pippis |
| 2 | Sara | Ricci |
| 3 | Marco | Depaolis |

Tutti gli esempi di sopra sono stati realizzati usando la libreria "mysql" comune anche alle vecchie versioni del PHP, mentre nella pagina seguente vedremo l'uso della libreria "mysqli".

PHP e MySQL: ESEMPI - USO DELLA LIBRERIA MYSQLI

Ora vediamo gli stessi esempi visti nella precedente pagina, usando però la nuova libreria del PHP 5: **mysqli**. La novità introdotta da **mysqli** sta sicuramente nel fatto che questa libreria può essere usata sia con un normale approccio procedurale (troviamo quindi le stesse funzioni di "mysql_") e sia con un approccio ad oggetti, in quanto nel PHP 5 viene introdotta la classe **mysqli**, che si occupa proprio della connessione e dell'esecuzione di query su di un database MySQL attraverso il PHP, e della classe **mysqli_result** che si occupa invece della gestione dei risultati di una query. Scorrendo quindi le funzioni native e procedurali del PHP possiamo trovare una corrispondenza con le proprietà di queste classi; per cui alla vecchia funzione "mysql_query", ora corrisponde la funzione "mysqli_query" e il metodo equivalente "mysqli::query".

Per questo motivo di seguito sono mostrati gli stessi esempi sia con un approccio "procedurale" e sia con un approccio ad oggetti.

Approccio procedurale

Di seguito è mostrato l'uso delle funzioni necessarie per la connessione e relativa chiusura.

```
<?php
$connessione = mysqli_connect("localhost","root","","dbtest") or die("Connessione non riuscita: ".mysqli_connect_error());
print("Identificativo di Connessione : ".$connessione);
```

```
mysqli_close($connessione);
?>
```

Di seguito, invece, è mostrato un altro esempio: supponendo di avere il solito database *rubrica*, con la tabella *nomi* contenente appunto solo i nomi, si vuole printare l'elenco di tutti i nomi presenti nel database.

```
<?php
$connessione = mysqli_connect("localhost","root","","rubrica");
if (mysqli_connect_errno()) {
    echo("Errore durante la connessione al server MySQL");
    exit();
}
else {
    echo("Connessione effettuata con successo");
}
$resultato = mysqli_query("SELECT * FROM nomi",$connessione);
while($riga=mysqli_fetch_assoc($resultato))
{
    echo("ID: ".$riga['id']." NOME: ".$riga['nome']."<br>");
}
mysqli_free_result($resultato);
mysqli_close($connessione);
?>
```

Da questi esempi, possiamo vedere che le funzioni sono essenzialmente le stesse di quelle della libreria *mysql*, con l'unica differenza sostanziale che *mysqli_connect* riceve come parametro anche il nome del database e l'eventuale porta di connessione e la funzione *mysqli_fetch_assoc* ritorna direttamente un array "associato" (cioè contenente i nomi dei campi come chiavi e come valori di questi, i valori effettivi contenuti nel database).

Approccio procedurale

Vediamo direttamente l'ultimo esempio più completo, introducendo il concetto di classe o oggetto (vi rimando alla [pagina](#) della guida per approfondimenti in merito).

```
<?php
$mysqli = new mysqli("localhost","root","","rubrica");
if (mysqli_connect_errno()) {
    echo("Errore durante la connessione al server MySQL");
    exit();
}
else {
    echo("Connessione effettuata con successo");
}
$resultato = $mysqli->query("SELECT * FROM nomi");
while($riga = $resultato->fetch_assoc())
{
    echo("ID: ".$riga['id']." NOME: ".$riga['nome']."<br>");
}
$resultato->close();
$mysqli->close();
?>
```

Da notare il differente modo di richiamare le funzioni, tipico proprio di un approccio ad oggetti, identificato dalla prima riga del codice (istanza della classe *mysqli*). Altra differenza è il modo in cui viene liberata la memoria dalla variabile *\$resultato*: la funzione *close()* corrisponde infatti alla vecchia *mysqli_free_result()*.