2017

MY SQL

Guida MySQL di base

Progetto di Alternanza Scuola-Lavoro – Anno scolastico 2016-2017 Istituto Tecnico Enrico Fermi, Siracusa

SikeWEB

Click different

Eleonora
SikeWEB
15/05/2017

MySQL

Collegarsi ad un database e interagire con i dati in esso archiviati è indubbiamente uno dei compiti principali di un sito Web dinamico costruito con PHP. Lo sviluppatore interessato a costruire delle applicazioni che possano colloquiare con MySQL, il DBMS (Database Management System) Open Source per la gestione di database relazionali più utilizzato in Rete con PHP, ha a sua disposizione l'estensione **MySQLi** (*MySQL improved*), ossia un'interfaccia ad oggetti con connessione al database.

MySQLi

Istanziare la classe per connettersi al database

Per iniziare a lavorare con la libreria è necessario innanzitutto istanziare la relativa classe: il costruttore di MySQLi utilizzerà i parametri da noi forniti, o quelli di default se non ne vengono specificati altri, per aprire la connessione con il DBMS, essa sarà il nostro tramite per comunicare di volta in volta le operazioni da svolgere sul database. I metodi connect_error() e connect_errno() potrebbero esserci utili per controllare l'avvenuta connessione:

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

Questo semplice codice apre la connessione creando un nuovo oggetto, e poi la verifica tramite il metodo connect_error(): nel caso si verifichi un errore lo script viene interrotto stampando a schermo il codice di errore (connect_errno) e la sua descrizione, mentre se la connessione viene stabilita viene invocato il metodo host_info che restituisce una stringa con l'host del DBMS e il metodo di connessione.

I parametri da passare al costruttore sono soltanto 4 più un argomento opzionale:

- l'host, cioè l'indirizzo della macchina in cui sta girando MySQL, indicato tramite il nome o l'indirizzo IP (in locale nella stragrande maggioranza dei casi si tratterà di localhost);
- il nome utente dell'utilizzatore abilitato ad inviare istruzioni al DBMS sulla base dei permessi che gli sono stati accordati;
- la password associata al nome utente utilizzato;
- il nome del database;
- eventualmente la porta (di base impostata su 3306) e il socket (È il punto in cui un processo accede al canale di comunicazione per mezzo di una porta)

MySQLi: creazione del database e delle tabelle

Creiamo quindi la connessione a DBMS, ma stavolta senza specificare il nome del database perché vogliamo crearne uno apposito per l'applicazione, poi utilizziamo uno dei metodi più importanti della classe, query:

```
<?php
$mysqli = new mysqli('localhost', 'root', '');
   if ($mysqli->connect_error) {
      die('Errore di connessione ('.$mysqli->connect_errno.') '.$mysqli->connect_error);
}
// Creo il database
$mysqli->query("CREATE DATABASE biblioteca");
// Seleziono il database
$mysqli->query("USE biblioteca");
?>
```

I parametri inseriti sono:

- host=localhost
- username=root
- password= campo vuoto

Il metodo query accetta come parametro la query SQL da passare a MySQL, e restituisce il booleano TRUE a meno che la query generi un errore. Nel codice proposto l'abbiamo utilizzato prima per creare il database dell'applicazione e poi per selezionare quest'ultimo come nostro ambiente di lavoro.

Il metodo error può essere usato di seguito per ottenere la stringa di errore eventualmente generata da MySQL. Supponiamo ad esempio di scrivere:

```
<?php
$mysqli = new mysqli('localhost', 'root', '');
    if ($mysqli->connect_error) {
        die('Errore di connessione ('.$mysqli->connect_errno.') '.$mysqli-
>connect_error);
}
// Creo il database
if (!$mysqli->query("CRREATE DATABASE biblioteca")){
    die($mysqli->error);
}
// Seleziono il database
$mysqli->query("USE biblioteca");
?>
```

In questo caso si tratta di un semplice errore di battitura del comando CREATE. Otterremo quindi come risultato l'arresto dello script e la stampa a schermo dell'errore restituito da MySQL.

La creazione delle tabelle

Procediamo sempre col metodo query per la creazione delle tabelle necessarie, e richiamando la keyword SQL CREATE:

```
// Seleziono il database
$mysqli->query("USE biblioteca");
// creazione della tabella per il login
$mysqli->query("CREATE TABLE login
                          ( id INT ( 5 ) NOT NULL AUTO_INCREMENT,
                            user VARCHAR(40) NOT NULL,
                            password VARCHAR(64) NOT NULL,
                            PRIMARY KEY (id))");
// creazione della tabella per i libri
$mysqli->query("CREATE TABLE libri
                          ( id INT(5) NOT NULL AUTO_INCREMENT,
                            autore VARCHAR(40) NOT NULL,
                            titolo TEXT NOT NULL,
                            editore VARCHAR(40) NOT NULL,
                            anno SMALLINT(2) NOT NULL,
                            PRIMARY KEY (id))");
```

Si noti che l'id di ogni tabella è impostato come un valore univoco automaticamente incrementato dal DBMS. Vediamo ora nel dettaglio quali campi abbiamo creato per ciascuna tabella. Prevediamo naturalmente l'esistenza di un backend, ed è quindi necessario predisporre una tabella per gestire le credenziali di accesso. Per la tabella login abbiamo impostato tre campi:

Campo	Descrizione
id	Un campo numerico che funge da chiave primaria, che verrà incrementato automaticamente da MySQL ad ogni nuovo inserimento (lo inseriremo in ogni tabella).
user	Un campo alfanumerico per il nome dell'utente, di lunghezza massima di 40 caratteri.
password	Un campo alfanumerico di 64 caratteri per la password dell'utente. Nello specifico non verrà memorizzata la password, bensì il suo hash di 64 caratteri generato tramite l'algoritmo SHA256.

Per la tabella dei **libri**, in pratica il nostro catalogo, abbiamo costruito questa struttura:

Campo	Descrizione
id	Vale quanto già specificato per tutte le tabelle.
autore	Un campo alfanumerico di 40 caratteri per l'autore del libro.
editore	Un campo alfanumerico di 40 caratteri per l'editore del libro.
titolo	Un campo testuale dalla lunghezza non definita per il titolo.
anno	Un campo numerico per l'anno di pubblicazione. Ha un range molto ridotto per il quale è sufficiente il tipo smallint.

All'interno della nostra applicazione dovremo svolgere diversi compiti tramite i dati con i quali popoleremo il database. Useremo ad esempio la tabella login per verificare le credenziali degli operatori;

MySQLi: inserimento e lettura dei record

Dopo aver creato la struttura delle tabelle passiamo ad analizzare l'inserimento e la lettura dei dati tramite la libreria MySQLi. È naturale per un'applicazione Web con un'interfaccia di backend verificare l'accesso degli operatori autorizzati tramite le credenziali memorizzate in una apposita tabella di login. Per quest'ultima, lo ricordiamo, abbiamo predisposto i campi id, user e password. Nel campo password non inseriremo comunque del testo in chiaro, ma l'hash di 64 caratteri della password generato tramite la funzione SHA256 disponibile nativamente in PHP.

Inserimento dei dati nelle tabelle

L'inserimento dei dati avviene tramite query SQL con la parola chiave INSERT processata da MySQLi tramite il metodo **query():**

```
// Dopo la connessione al database e la creazione di eventuali tabelle

$password = hash('sha256', 'paperino'); //Creazione dell'hash
$query = "INSERT INTO login (user, password) VALUES ('pippo', '$password')";
// Esecuzione della query e controllo degli eventuali errori
if (!$mysqli->query($query)) {
    die($mysqli->error);
}
```

Abbiamo detto in precedenza che il metodo **query()** restituisce il valore false nel caso in cui MySQL risponda alla richiesta con un messaggio di errore per il debug. Non è tuttavia l'unico dato che possiamo ricavare dopo l'esecuzione della query, altre due informazioni che potrebbero risultare molto utili sono il numero delle righe generate e, trattandosi di un inserimento all'interno di una tabella con una campo autoincrement, l'ultimo ID inserito.

Per testare il semplice funzionamento dei metodi **affected_rows()** e **insert_id(),** applicabili all'istanza della classe mysqli_result ottenuta come risultato dal metodo query, al termine del codice possiamo aggiungere queste righe:

```
echo "Righe generate: " .  $mysqli->affected_rows . "<br />";
echo "Ultimo ID inserito: " . $mysqli->insert_id  . "<br />";
```

Verifica delle credenziali

Per verificare la correttezza delle credenziali dobbiamo passare alla lettura dei dati, utilizzando come parametri quelli inseriti dall'utente verosimilmente da un form. Un esempio di codice potrebbe essere il seguente:

```
$user = "pippo"; // Normalmente questi valori sarebbero ottenuti tramite POST
$password = hash('sha256', "paperino");

$query = $mysqli->query("SELECT * FROM login WHERE user = '$user' AND password =
'$password'");
if($query->num_rows) {
    echo "Accesso consentito";
} else {
    echo "Accesso rifiutato";
}
```

Il metodo **num_rows(),** sempre applicabile alla classe mysqli_result, restituisce il numero di righe ottenute dalla select e avremo quindi un risultato superiore a 0 solo nel caso in cui le credenziali siano valide.

Update dei record

Per aggiornare il valore di un campo all'interno di una tabella utilizzeremo UPDATE

```
$query = $mysqli->query("UPDATE login SET user='pluto',password='plutone' WHERE id=1");
```

Con Where andremo a settare la condizione verificata la quale verrà eseguita la query di aggiornamento.

Eliminazione dei record

Per eliminare un record da una tabella utilizzeremo DELETE

```
$query = $mysqli->query("DELETE FROM login WHERE id = 1");
```

Mysqli: fetch. Recuperare i risultati di una query

Facciamo un semplice esempio immaginando di usare la nostra solita tabella e avendo la connessione al database già a nostra disposizione. Come prima operazione definiamo la query SQL e andiamo ad eseguire la nostra query:

```
$query = $mysqli->query("SELECT user, password FROM login");
```

A questo punto si pone il problema di recuperare i dati nella nostra pagina, la prima soluzione sarà:

```
$row = $query->fetch_array(MYSQLI_NUM);
```

potrebbe essere comodo fare la stampa ricorsiva dell'array $row:

```
print_r($row);
```

se poi inseriamo tutto in un ciclo while abbiamo a disposizione l'intero set dei risultati:

```
$query = $mysqli->query("SELECT user, password FROM login");
while($row = $query->fetch_array(MYSQLI_NUM)){
   print_r($row);
}
```

Esempio di controllo utente registrato

Supponiamo di avere la tabella login all'interno del nostro database, e di voler verificare che l'utente che ha compilato il campo della nostra form e di cui conosco user e password sia presente.

Ricordiamo inoltre che la password inserita in fase di registrazione è stata salvata sul DB criptata.

```
$user = $_POST['username'];// Normalmente questi valori sono ottenuti tramite POST dalla form
$password = $_POST['password'];
$password_criptata=hash('sha256', $password);

$query = $mysqli->query("SELECT user, password FROM login WHERE user= '$user' AND password =
'$password_criptata'");
if($query->num_rows) {
   echo "Utente  $user con pass: $password_criptata presente nella tabella di Login";
} else {
   echo "Utente non presente";
}
```