

## **Indice generale**

CREA DATABASE crea.php.....	1
APRI DATABASE apriDB-php e connessione.php.....	3
Creazione di una nuova tabella MySQL.....	4
Popolare la tabella con INSERT INTO.....	6
Selezione dei record con PHP e MySQLi.....	8
Modificare un record con MySQLi.....	10
Cancellare un record con MySQLi.....	12

# Crea database crea.php

L'estensione MySQLi è stata introdotta in PHP con la versione 5.0.0, mentre a partire dalla release 5.3 del linguaggio tale libreria è stata attivata e utilizza di default il **MySQL Native Driver**. Non è quindi richiesta alcuna ulteriore operazione per la sua abilitazione.

Per consentire la connessione tra applicazione e DBMS sarà necessario disporre anche in questo caso dei parametri richiesti, e glio:

- 1.host, cioè l'indirizzo della postazione che ospita l'installazione di MySQL, generalmente esso è "localhost" (soprattutto nel caso di script sviluppati nella macchina locale), ma può essere anche un nome di dominio o un indirizzo IP;
- 2.username: il nome dell'utente che possiede i permessi per effettuare operazioni che prevedano l'interazione con il DBMS e i dati da esso gestiti;
- 3.password: la parola chiave che l'utente precedentemente citato dovrà utilizzare per autenticarsi su MySQL.

```
crea.php x connessione.php x creaTabella.php x insertInto.php x selezione.php x update.php x delete.p
1  <?php
2  /*
3   creazione di un database con MySQLi.
4   La prima operazione richiesta sarà quella relativa alla definizione
5   del blocco dei parametri per la connessione
6  */
7  // nome di host
8  $host = "localhost";
9  // username dell'utente in connessione
10 $user = "root";
11 // password dell'utente
12 $password = "";
13
14 // stringa di connessione al DBMS
15 $connessione = new mysqli($host, $user, $password);
16
17 // verifica su eventuali errori di connessione
18 if ($connessione->connect_errno) {
19     echo "Connessione fallita: ". $connessione->connect_error . ".";
20     exit();
21 }
22
23 // esecuzione della query per la creazione del database
24 if (!$connessione->query("CREATE DATABASE nuova_rubrica")) {
25     echo "Errore della query: " . $connessione->error . ".";
26 }else{
27     echo "Database creato correttamente.";
28 }
29
30 // chiusura della connessione
31 $connessione->close();
32 ?>
33
```

Tali informazioni potranno essere utilizzate per la creazione di una stringa di connessione rappresentata dall'istanza dell'oggetto di classe **mysqli**; nel caso specifico l'applicazione controlla tramite il costrutto condizionale if/else e l'utilizzo dei metodi **connect\_errno()** e **connect\_error()** l'eventuale verificarsi di errori in fase di connessione. Nel caso in cui si dovessero verificare dei problemi, il primo metodo restituirà il codice identificativo dell'errore prodotto dall'ultimo tentativo di connessione effettuato, mentre il secondo metterà a disposizione una stringa descrittiva dell'errore generato.

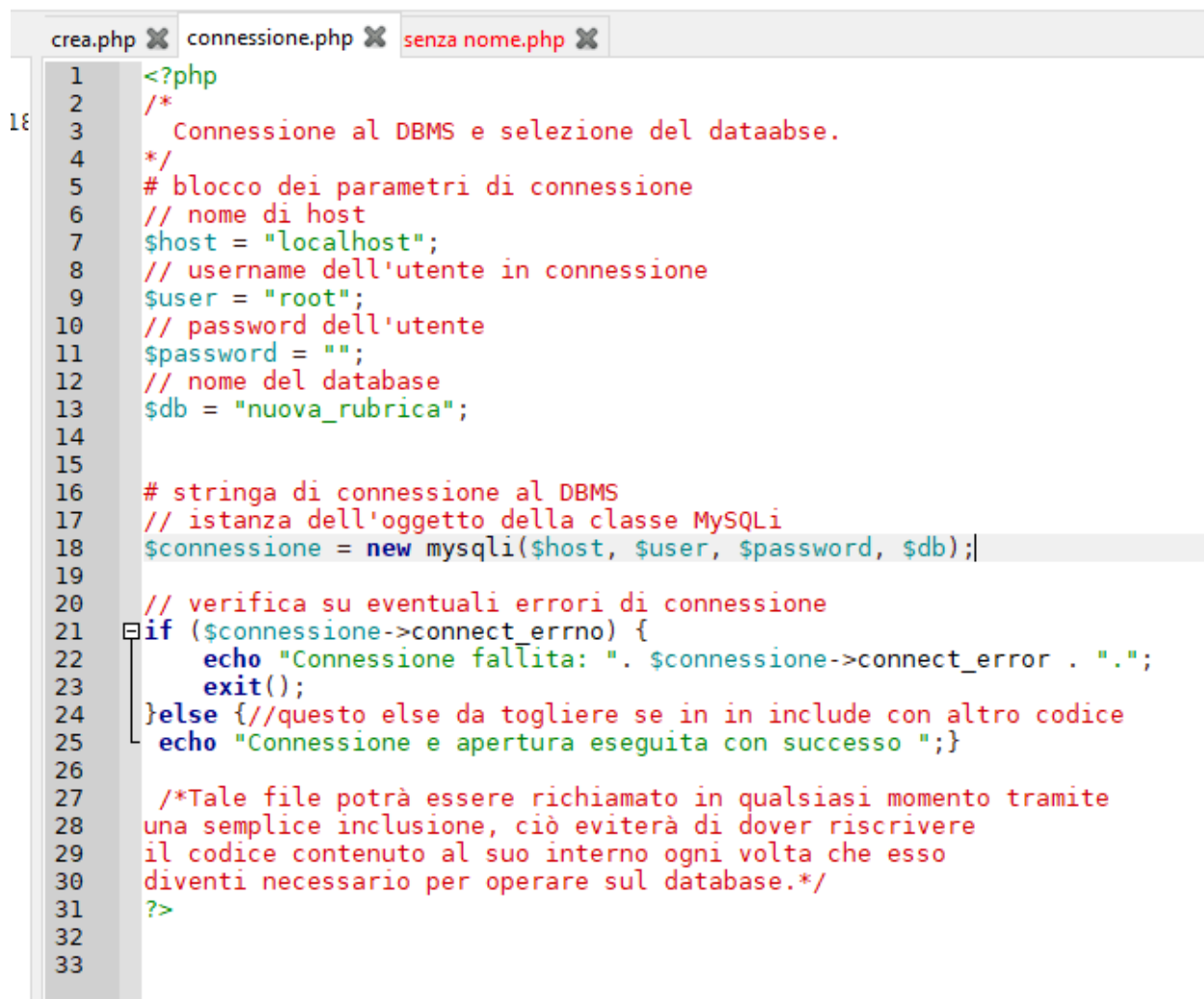
Nel caso in cui, invece, non dovessero presentarsi degli errori in connessione, l'applicazione proseguirà nel suo funzionamento passando alla fase relativa alla creazione del database, chiamato nel nostro caso "nuova\_rubrica", che potrà essere eseguita tramite il passaggio al metodo **query()** dell'istruzione SQL **CREATE DATABASE** seguita dal nome del database desiderato.

Indipendentemente dall'esito di quest'ultimo passaggio la connessione verrà chiusa grazie all'invocazione del metodo **close()**.

## APRI DATABASE apriDB.php e connessione.php

Ora che si dispone di un database su cui operare, sarà possibile passare all'istanza dell'oggetto di classe mysqli un ulteriore parametro, quello relativo al nome del database stesso; in questo modo sarà possibile ottenere un sorgente completo per la connessione e la selezione della base di dati che potrà essere salvato in un file (denominato ad esempio "connessione.php")

Tale file potrà essere richiamato in qualsiasi momento tramite una semplice inclusione, ciò eviterà di dover riscrivere il codice contenuto al suo interno ogni volta che esso diventi necessario per operare sul database.

The image shows a code editor with three tabs at the top: 'crea.php', 'connessione.php' (which is active), and 'senza nome.php'. The code in the active tab is a PHP script for connecting to a MySQL database. It includes comments in Italian explaining the purpose of the file and the steps taken. The code sets host, username, password, and database name, then creates a new mysqli object and checks for connection errors. A final comment explains that this file can be included in other scripts to avoid rewriting the connection code.

```
1 <?php
2 /*
3  * Connessione al DBMS e selezione del dataabse.
4  */
5 # blocco dei parametri di connessione
6 // nome di host
7 $host = "localhost";
8 // username dell'utente in connessione
9 $user = "root";
10 // password dell'utente
11 $password = "";
12 // nome del database
13 $db = "nuova_rubrica";
14
15
16 # stringa di connessione al DBMS
17 // istanza dell'oggetto della classe MySQLi
18 $connessione = new mysqli($host, $user, $password, $db);
19
20 // verifica su eventuali errori di connessione
21 if ($connessione->connect_errno) {
22     echo "Connessione fallita: ". $connessione->connect_error . ".";
23     exit();
24 }else {//questo else da togliere se in in include con altro codice
25     echo "Connessione e apertura eseguita con successo ";}
26
27 /*Tale file potrà essere richiamato in qualsiasi momento tramite
28 una semplice inclusione, ciò eviterà di dover riscrivere
29 il codice contenuto al suo interno ogni volta che esso
30 diventi necessario per operare sul database.*/
31 ?>
32
33
```

# Creazione di una nuova tabella MySQL

Il prossimo passaggio da effettuare sarà quindi quello di generare una tabella all'interno della base di dati, tale tabella sarà poi popolata tramite dei campi destinati ad ospitare le informazioni da archiviare.

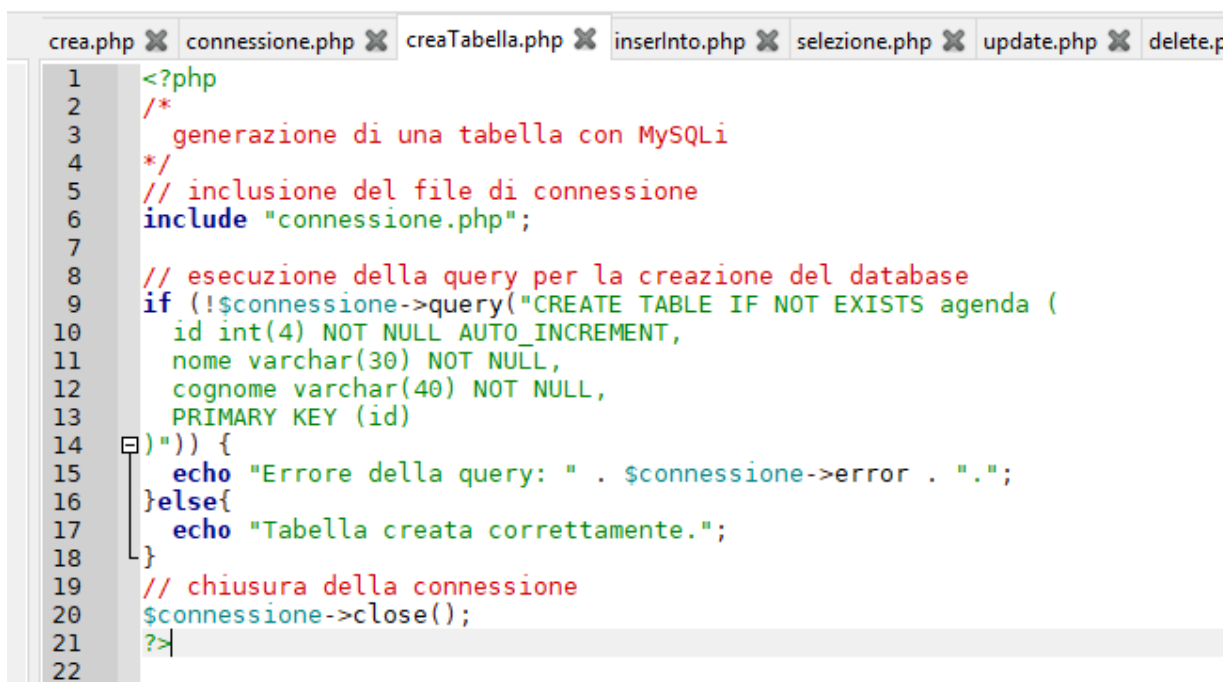
Nello specifico, la procedura proposta riguarderà la creazione di una tabella contenente 3 diversi campi:

- 1.id: un campo autoincrementale numerico di tipo intero della lunghezza di 4 cifre destinato a fungere anche da chiave primaria;
- 2.nome: un campo non nullo di tipo VARCHAR, destinato quindi ad ospitare valori alfanumerici, della lunghezza massima di 30 caratteri;
- 3.cognome: un campo non nullo di tipo VARCHAR della lunghezza massima di 40 caratteri.

Una volta definiti i campi di nostro interesse, questi potranno essere generati dopo che l'applicazione deputata a tale compito avrà effettuato la connessione a MySQL per poi selezionare il database da manipolare; come anticipato, tali operazione non dovranno essere ripetute perché già previste nel file "connessione.php" definito nel capitolo precedente e richiamato all'inizio del sorgente proposto di seguito attraverso l'istruzione **include()**

**ATTENZIONE** meglio utilizzare l'istruzione **require** che rispetto ad include crea un errore e non esegue le istruzioni successive.

**require\_once** inserisce codice se non già presente utilizzarlo per la selezione.



```
1  <?php
2  /*
3   generazione di una tabella con MySQLi
4  */
5  // inclusione del file di connessione
6  include "connessione.php";
7
8  // esecuzione della query per la creazione del database
9  if (!$connessione->query("CREATE TABLE IF NOT EXISTS agenda (
10     id int(4) NOT NULL AUTO_INCREMENT,
11     nome varchar(30) NOT NULL,
12     cognome varchar(40) NOT NULL,
13     PRIMARY KEY (id)
14 )")) {
15     echo "Errore della query: " . $connessione->error . ".";
16 }else{
17     echo "Tabella creata correttamente.";
18 }
19 // chiusura della connessione
20 $connessione->close();
21 ?>
```

Nel codice presentato l'istruzione SQL necessaria per la generazione della tabella viene passata come parametro al metodo query() che, come visto in precedenza, viene utilizzato per effettuare una richiesta ("query") ad un database; in

questo caso il comando SQL di riferimento è CREATE TABLE.

Grazie alla clausola IF NOT EXISTS, e al fine di evitare ambiguità, l'istruzione controllerà innanzitutto che il nome scelto per la tabella desiderata (chiamata in questo caso "agenda") non sia già stato utilizzato all'interno del database "nuova\_rubrica" per un'altra tabella, fatto questo e in mancanza di omonimie, si passerà alla fase relativa alla creazione dei campi.

Il metodo query() è delimitato all'interno di un blocco if/else, nel caso in cui l'istruzione SQL passata come parametro ad esso dia origine ad un errore, quest'ultimo verrà intercettato attraverso il metodo **error()**, esso infatti avrà il compito di restituire a video una stringa destinata a descrivere la natura dell'errore prodotto in seguito alla chiamata più recente ad una funzione di MySQLi.

Una volta eseguita la query al database, e indipendentemente dal risultato ottenuto, il già citato metodo close() si occuperà di chiudere la connessione al DBMS in modo da terminare il ciclo di esecuzione dello script.

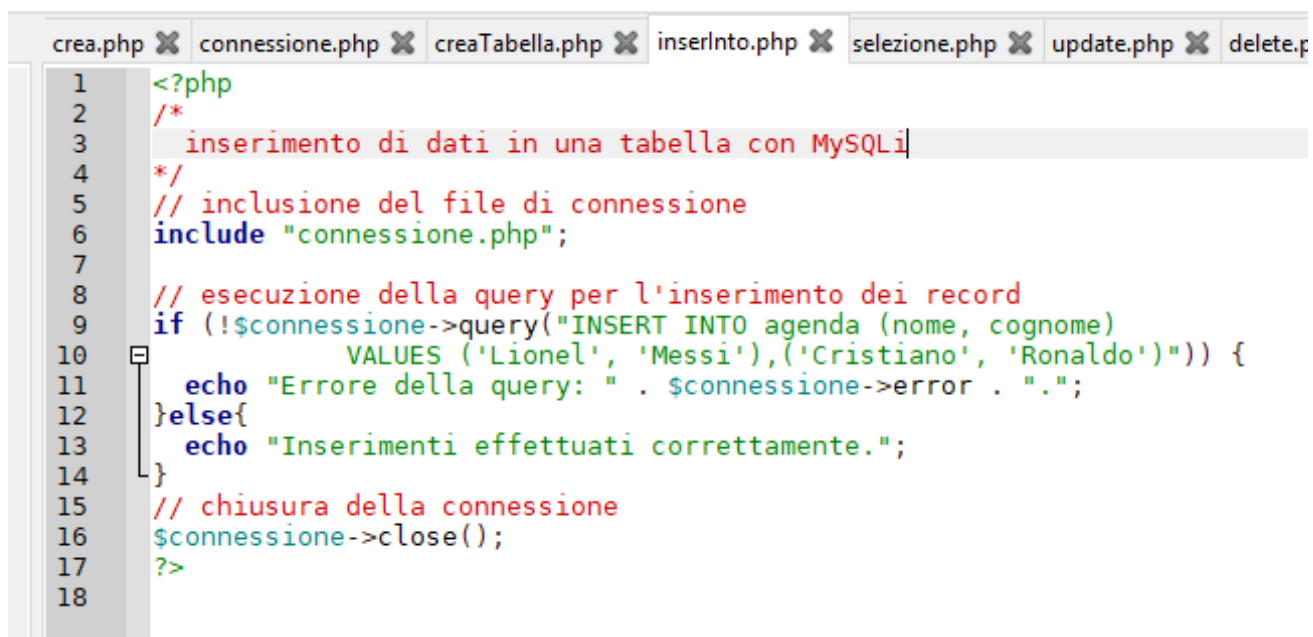
Se l'istruzione lanciata dovesse permettere di ottenere il risultato atteso, allora si potrà disporre di una tabella completa di campi che, a loro volta, potranno essere popolati tramite i dati.

## Popolare la tabella con INSERT INTO

Una volta creato il database ("nuova\_rubrica"), generata una tabella all'interno di esso ("agenda") con tanto di campi per la memorizzazione delle informazioni, la fase relativa all'inserimento dei dati apparirà particolarmente semplice; in questo caso infatti verrà fatto ricorso a metodi già ampiamente utilizzati in questa porzione di guida dedicata a MySQLi, a cambiare sarà invece l'istruzione SQL utilizzata.

Anche per quanto riguarda la procedura di archiviazione dei dati in tabella non si potrà fare a meno di includere il file "connessione.php" (definito nel in una lezione precedente) per effettuare i passaggi relativi alla connessione al Database Manager e alla selezione del database da utilizzare; ciò avverrà quindi, come visto in precedenza, all'inizio del codice.

Nel caso specifico dell'esempio proposto di seguito verranno inseriti due nuovi valori in corrispondenza dei campi "nome" e "cognome", il campo "id", essendo un autoincrementale, provvederà ad incrementarsi autonomamente e automaticamente in concomitanza con l'aggiunta dei due nuovi record. Gli approfondimenti su i diversi passaggi verranno forniti immediatamente dopo il sorgente.



```
crea.php X connessione.php X creaTabella.php X insertInto.php X selezione.php X update.php X delete.p
1  <?php
2  /*
3   inserimento di dati in una tabella con MySQLi
4  */
5  // inclusione del file di connessione
6  include "connessione.php";
7
8  // esecuzione della query per l'inserimento dei record
9  if (!$connessione->query("INSERT INTO agenda (nome, cognome)
10     VALUES ('Lionel', 'Messi'),('Cristiano', 'Ronaldo')")) {
11     echo "Errore della query: " . $connessione->error . ".";
12 }else{
13     echo "Inserimenti effettuati correttamente.";
14 }
15 // chiusura della connessione
16 $connessione->close();
17 ?>
18
```

Come anticipato, le novità contenute nel codice precedente non sono tantissime; sostanzialmente, dopo l'inclusione del file per la connessione al DBMS e la selezione del database verrà richiamato il già noto metodo query(). Quest'ultimo, come già più volte visto, si occuperà di eseguire un'istruzione SQL basata sul comando INSERT INTO a cui seguirà il nome della tabella coinvolta, i nomi dei campi da popolare (indicati tra parentesi tonde) e i valori da inserire introdotti tramite la clausola VALUES.

Anche questa volta il metodo per l'esecuzione della query è interno ad un costrutto condizionale, infatti, grazie all'utilizzo del blocco if/else l'applicazione avrà la possibilità di controllare l'eventuale presenza di errori collegati

all'esecuzione dell'istruzione SQL; il metodo `error()` si occuperà quindi di restituire a video una breve descrizione di tale errore nel caso in cui esso dovesse verificarsi venendo intercettato.

Se non dovessero essere generati errori, lo script proseguirà nella sua esecuzione confermando l'avvenuto inserimento; vi sarà poi il passaggio alla fase di chiusura della connessione che avverrà indipendentemente dal risultato (e dal funzionamento) della query tramite l'azione del metodo `close()`.

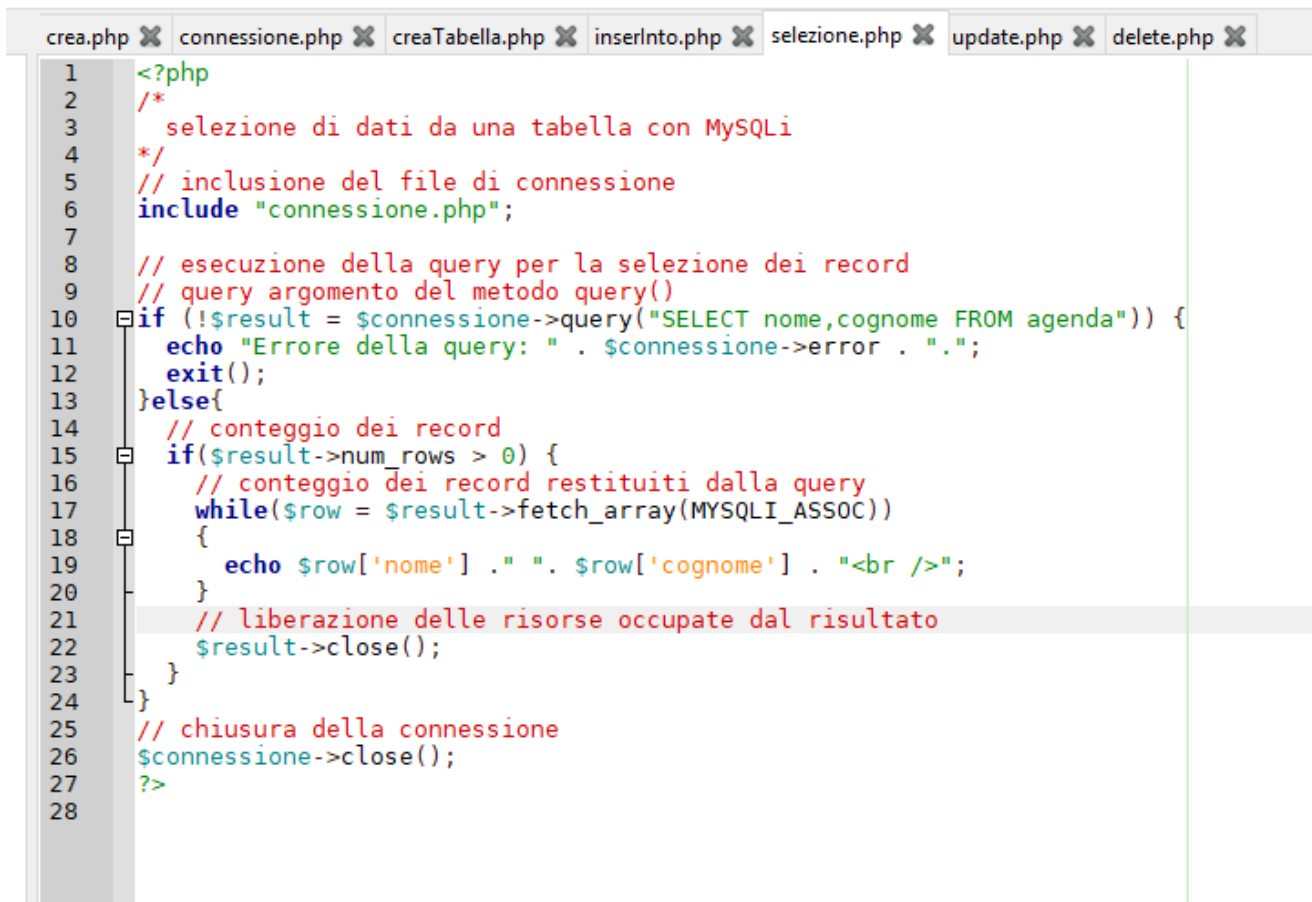
Anche se non particolarmente articolata, almeno per quanto riguarda il semplice caso proposto, la procedura richiesta per l'inserimento dei dati è fondamentale perché mette a disposizione delle informazioni che potranno essere estratte, aggiornate o cancellate (si ricordino i precedenti riferimenti allo schema CRUD, *Create*, *Read*, *Update* e *Delete*, per la gestione dei dati), come accadrà nei capitoli seguenti tramite l'utilizzo delle funzionalità messe a disposizione da MySQLi.



# Selezione dei record con PHP e MySQLi

Ora che la tabella "agenda" del database "nuova\_rubrica" presenta dei campi popolati con dei record, è possibile passare alla fase di estrazione dei dati con successiva stampa a video; per far questo l'applicazione basata su MySQLi proposta di seguito si occuperà di interrogare il Database Manager richiedendo l'esecuzione di una query basata sul comando SELECT.

Nel caso specifico dell'esempio mostrato di seguito verranno estratti tutti i dati fino ad ora memorizzati in tabella relativamente ai campi "nome" e "cognome", ciò avverrà però soltanto dopo aver verificato tramite un apposito controllo che quest'ultima presenti effettivamente dei record.



```
1 <?php
2 /*
3  * selezione di dati da una tabella con MySQLi
4  */
5 // inclusione del file di connessione
6 include "connessione.php";
7
8 // esecuzione della query per la selezione dei record
9 // query argomento del metodo query()
10 if (!$result = $connessione->query("SELECT nome,cognome FROM agenda")) {
11     echo "Errore della query: " . $connessione->error . ".";
12     exit();
13 }else{
14     // conteggio dei record
15     if($result->num_rows > 0) {
16         // conteggio dei record restituiti dalla query
17         while($row = $result->fetch_array(MYSQLI_ASSOC))
18         {
19             echo $row['nome'] . " " . $row['cognome'] . "<br />";
20         }
21         // liberazione delle risorse occupate dal risultato
22         $result->close();
23     }
24 }
25 // chiusura della connessione
26 $connessione->close();
27 ?>
28
```

Come già accaduto in precedenza, anche nell'esempio appena mostrato l'istruzione per l'interrogazione al DBMS è stata passata come argomento al metodo query();

nel caso in cui tale operazione dovesse dare luogo a degli errori, allora il metodo error() consentirà di stampare a video una stringa descrittiva riguardante la natura dell'errore stesso.

Il verificarsi di un errore interromperà l'esecuzione dello script grazie alla funzione **exit()**, in caso contrario, cioè se la query dovesse essere eseguita correttamente, l'applicazione richiamerà il metodo **num\_rows()** che avrà il compito di

contare i record coinvolti da quest'ultima. Nel caso in cui il conteggio effettuato dovesse restituire un risultato superiore a zero:

```
if($result->num_rows > 0) { ...
```

allora si potrà passare alla fase di stampa dei dati estratti; essa sarà possibile tramite un ciclo while le cui iterazioni saranno pari al numero dei record selezionati tramite l'istruzione SQL.

Per ogni iterazione del ciclo verrà richiamato il metodo **fetch\_array()**, esso è stato concepito per riportare un set di risultati sotto forma di array associativo, numerico o dotato di entrambe tali caratteristiche; a tal proposito è disponibile un apposito parametro sotto forma di costante in grado di definire quale tipo di array dovrà essere prodotto tramite l'esecuzione del metodo.

Tale parametro potrà assumere tre diversi valori: **MYSQLI\_ASSOC** (array associativo, come nel caso dell'esempio proposto), **MYSQLI\_NUM** (array numerico) o **MYSQLI\_BOTH** (entrambe le caratteristiche).

Il ciclo consentirà quindi di stampare uno per uno tutti i record estratti prelevandoli direttamente dall'array generato (nel nostro caso denominato "**\$row**"), i valori restituiti a video saranno quelli corrispondenti alle chiavi associative utilizzate (**\$row['nome']** e **\$row['cognome']**).

Fatto questo il metodo `close()` permetterà di liberare le risorse impiegate per la memorizzazione del risultato ottenuto e di interrompere la connessione a MySQL.

## Modificare un record con MySQLi

Una volta inseriti dei record all'interno della tabella "agenda", le informazioni archiviate in essa potranno essere aggiornate (e quindi modificate) o cancellate. Per fare ciò faremo ricorso, rispettivamente, alle istruzioni SQL **UPDATE** e **DELETE**.

A tal proposito, nel codice che verrà proposto di seguito saranno eseguite nell'ordine due operazioni distinte:

- la prima riguarderà appunto l'aggiornamento di alcuni dei dati precedentemente memorizzati,
- la seconda consisterà nell'estrazione del record coinvolto dall'UPDATE (ma potrebbero essere più

di uno) tramite query **SELECT** e nella successiva stampa a video delle informazioni modificate:

```
crea.php ✕ connessione.php ✕ creaTabella.php ✕ insertInto.php ✕ selezione.php ✕ update.php ✕ delete.php ✕
1  <?php
2  /*
3      upgrade di un record in una tabella con MySQLi
4  */
5  // inclusione del file di connessione
6  include "connessione.php";
7
8  // esecuzione della query per l'aggiornamento dei record
9  if (!$connessione->query("UPDATE agenda SET nome='Lio' WHERE id=1")) {
10     echo "Errore della query UPDATE: " . $connessione->error . ".";
11     exit();
12 }else{
13     // esecuzione della query per la selezione dei record
14     // query argomento del metodo query()
15     if (!$result = $connessione->query("SELECT nome,cognome
16                                         FROM agenda WHERE id=1")) {
17         echo "Errore della query: " . $connessione->error . ".";
18         exit();
19     }else{
20         // conteggio dei record
21         if($result->num_rows > 0) {
22             // ciclo i record restituiti dalla query
23             while($row = $result->fetch_array(MYSQLI_ASSOC))
24             {
25                 echo $row['nome'] . " " . $row['cognome'] . "<br />";
26             }
27             // liberazione delle risorse occupate dal risultato
28             $result->close();
29         }
30     }
31 }
32 // chiusura della connessione
33 $connessione->close();
34 ?>
35
```

Chiaramente, il primo passaggio da eseguire sarà quello relativo all'inclusione del file che contiene il codice necessario per la connessione al Database engine e la selezione del database ("config.php"); fatto questo si potrà lanciare immediatamente l'istruzione per l'aggiornamento del record che si desidera modificare:

```
"UPDATE agenda SET nome='Lio' WHERE id=1"
```

veloce, facile e sicuro.

```
"UPDATE agenda SET nome='Lio' WHERE id=1"
```

Tale istruzione verrà passata come argomento al **metodo query()** che si occuperà di eseguirla, nel caso in cui questa procedura dovesse dar luogo ad errori o comportamenti imprevisti, il **metodo error()** permetterà di visualizzare una notifica, prodotta da PHP stesso, indicante la natura dell'eccezione che dovrà essere gestita.

In presenza di errori, la funzione **exit()**, inserita come esito vincolante in caso di soddisfazione della condizione prevista nel costrutto di controllo if/else, impedirà che le istruzioni successive vengano eseguite.

Nel caso in cui l'UPDATE dovesse invece concludersi con successo, il **metodo num\_rows()** si occuperà di verificare che i record coinvolti dall'aggiornamento siano in numero maggiore a zero; un risultato pari a uno o superiore permetterà di lanciare un ciclo while che, per ogni sua iterazione, consentirà di richiamare il **metodo fetch\_array()** con cui prelevare l'array associativo (\$row) contenente le informazioni da restituire a video.

Al metodo fetch\_array() verrà passato un parametro associato al valore MYSQLI\_ASSOC che, come indicato nel capitolo precedente, permetterà di definire quale tipo di array dovrà essere prodotto dall'esecuzione del metodo.

Una volta effettuate le operazioni richieste, il **metodo close()** consentirà di liberare le risorse occupate dal risultato della SELECT e di terminare la connessione al DBMS MySQL.

## Cancellare un record con MySQLi

I dati di una tabella contenuta in un database possono essere cancellati tramite delle istruzioni SQL esattamente come possono esseri inseriti, estratti o modificati; anche in questo caso vale la pena ricordare che l'eliminazione di un record precedentemente archiviato rappresenta un'operazione definitiva, irreversibile: MySQL infatti non prevede un sistema per mantenere in memoria le informazioni rimosse.

Il comando SQL di riferimento per eseguire questo tipo di procedura è naturalmente **DELETE FROM**, ad esso andrà passato come argomento il nome della tabella coinvolta (nel caso del nostro esempio "agenda"), mentre la clausola WHERE permetterà di definire il record coinvolto o più record nel caso in cui si voglia procedere ad un maggior numero di cancellazioni.

```
1 <?php
2 /*
3  cancellazione di un record da una tabella con MySQLi
4  */
5  // inclusione del file di connessione
6  include "connessione.php";
7
8  // esecuzione della query per la selezione dei record
9  if (!$result = $connessione->query("DELETE FROM agenda WHERE id=2")) {
10     echo "Errore della query: " . $connessione->error . ".";
11     exit();
12 }
13
14 // chiusura della connessione
15 $connessione->close();
16 ?>
```

Come per qualsiasi altra operazione a carico del DBMS, anche in questo caso la fase relativa alla connessione a MySQL e quella riguardante la selezione del database dovranno precedere qualsiasi altra istruzione; motivo per il quale il primo passaggio da eseguire sarà quello di includere il file "config.php" contenente il codice necessario a questo scopo.

Fatto ciò si potrà procedere con l'esecuzione dell'istruzione SQL che, nel caso specifico, verrà effettuata a carico del record associato all'identificatore univoco ("id") "1":

```
"DELETE FROM agenda WHERE id=1"
```

Tale istruzione verrà introdotta all'interno di una struttura di controllo if/else, quest'ultima in pratica prevede che nel caso in cui il comando lanciato non possa essere eseguito, a causa di un errore nella sintassi o di un malfunzionamento, il metodo error() dovrà occuparsi di inviare una breve segnalazione in grado di descrivere la natura della problematica eventualmente verificatasi. Il già citato metodo exit() consentirà poi all'applicazione di arrestarsi in modo che alla generazione di un errore non faccia seguito alcuna esecuzione successiva.

In assenza di errori, l'istruzione lanciata avrà avuto plausibilmente esito positivo, motivo per il quale si potrà procedere con la chiamata al metodo `close()` attraverso il quale liberare le risorse impegnate e terminare la connessione al Database engine.