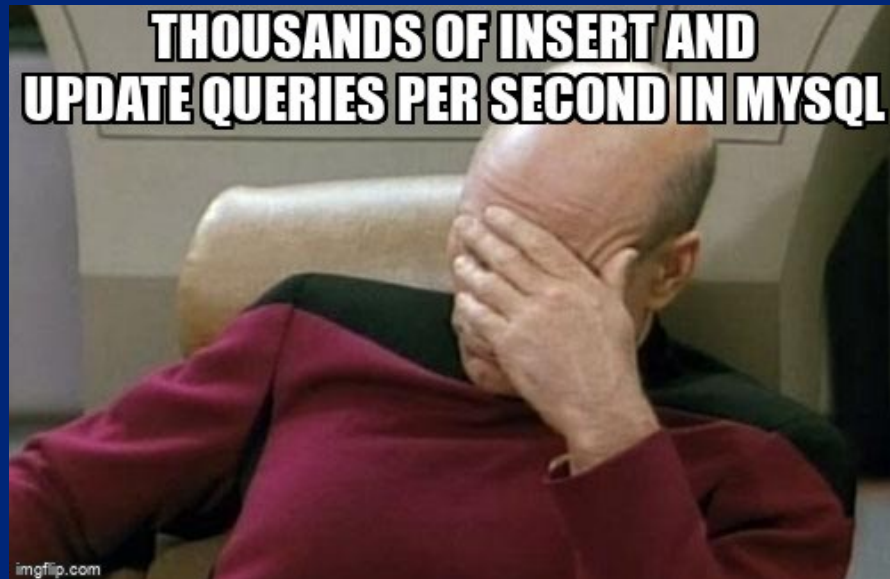


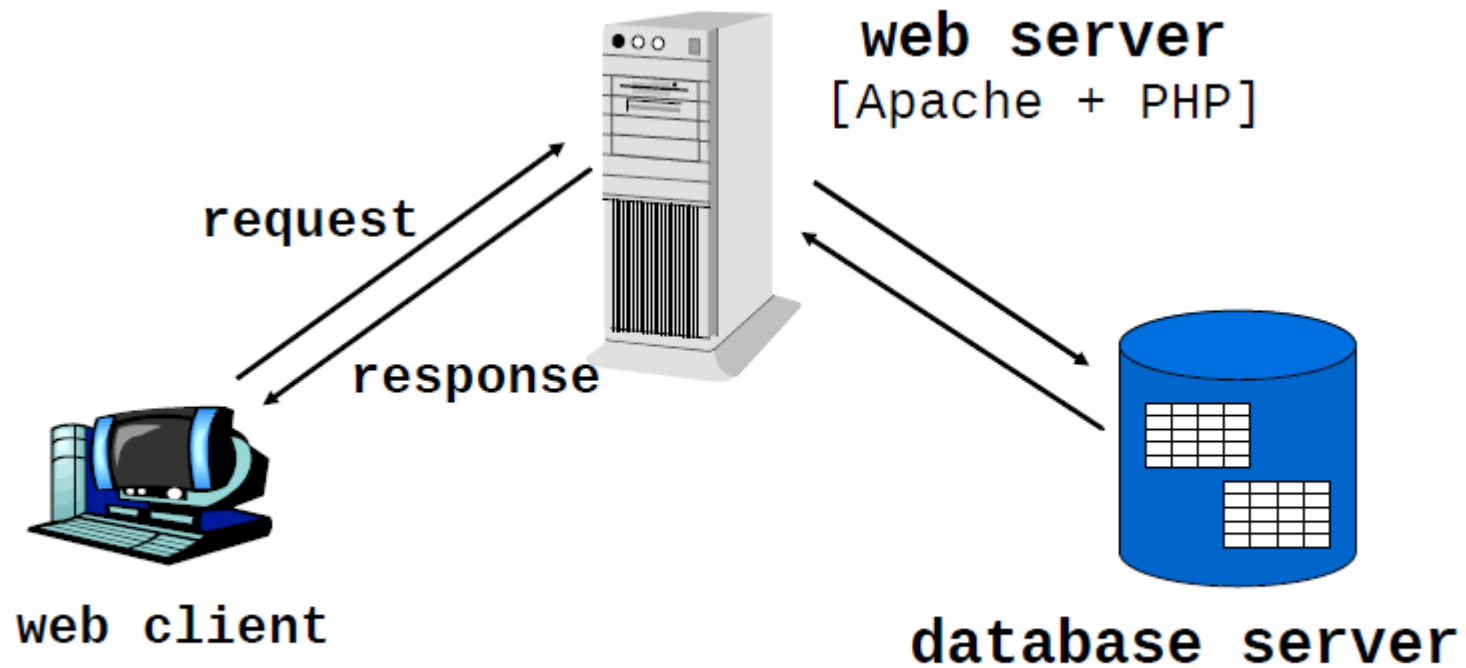
PHP (7)



Marina Ribaudò, marina.ribaudò@unige.it

Interazione con il database

2



Interagire con il database: how to

3

- Per accedere ad un database da una applicazione web, i passi fondamentali sono i seguenti
 1. Controllare i dati in arrivo dal client
 2. Stabilire una connessione e selezionare un database
 3. Preparare la query
 4. Eseguire la query
 5. Ottenere il risultato
 6. Formattare il risultato per l'utente
 7. [Chiudere la connessione]

PHP+MySQL: how to

4

- *There are two main API options when considering connecting to a MySQL database server:*
 - **PHP's mysqli Extension** (*"i" sta per improved*)
 - **PHP Data Objects (PDO)** (*database abstraction layer, permette di lavorare su database diversi senza cambiare il codice*)

1. Controllare i dati in arrivo

5

```
$nomevar = $_POST['...'];  
$nomevar = trim($nomevar);
```

```
mysqli_real_escape_string();
```

- **escapes special characters** (NUL, \n, \r, \, ', ", and CTRL+Z) for strings used in SQL statements
- takes into account the current charset of the **connection**
- helps protecting against SQL injection, it does not protect against other attacks like Cross-Site Scripting
- it is considered a **legacy approach**



<https://www.php.net/manual/en/mysqli.real-escape-string.php>

2. Stabilire la connessione

6

Stile procedurale

```
$con = mysqli_connect("localhost","name-of-user","password-of-user","db-name");  
if (mysqli_connect_errno()) {  
    // messaggio per il developer  
    echo "Failed to connect to MySQL: " . mysqli_connect_error();  
    // oppure messaggio per l'utente  
    echo "Something went wrong, visit us again later");  
}
```

Stile object oriented

```
$con = new mysqli("localhost","name-of-user","password-of-user","db-name");
```

Per il controllo dell'errore vedi:

<https://www.php.net/manual/en/mysqli.construct.php>

2. Stabilire la connessione

7

- Per stabilire la connessione bisogna “passare” al DBMS il nome del server, l’utente MySQL, la sua password, il nome del database
- Di solito **queste informazioni si mettono su un file a parte** condiviso da tutti gli script che hanno necessità di comunicare con il DBMS
- Se le informazioni sulla connessione vengono ripetute negli script PHP, cosa può capitare se si sposta il proprio sito web su un altro server



3. Preparare la query: SELECT

8

In questo primo esempio la query “mischia” codice SQL e dati in arrivo dal client



```
$query = "SELECT * FROM user WHERE name='$name' AND  
surname='$surname'";
```

Attenzione!

La query viene passata al database e bisogna ricordare l'**escape** dei dati di input (passo 1) per evitare errori o attacchi

1. Cosa succede con input **Dell'Amico**?
2. Cosa succede con input **Mario' OR 1=1; --** ?

3. Preparare la query: NOTE

9

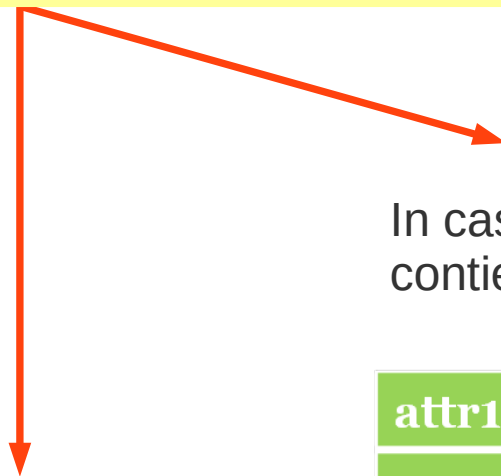
- Sebbene si possano usare i caratteri “...” per delimitare le stringhe, in SQL di solito si usano i caratteri ‘...’ (standard su tutti i DBMS)
- Bisogna fare attenzione all’**alternanza** tra “ e ‘ per distinguere correttamente le stringhe del codice PHP da quelle che fanno parte delle query SQL
- Per evitare SQL injection vedremo i **prepared statement**!

4. Eseguire la query

10

```
$res = mysqli_query($con,$query);
```

```
$res = $con->query($query);
```



In caso di **SELECT** viene restituita una **risorsa** che contiene i dati estratti dalla query

False in caso di errore

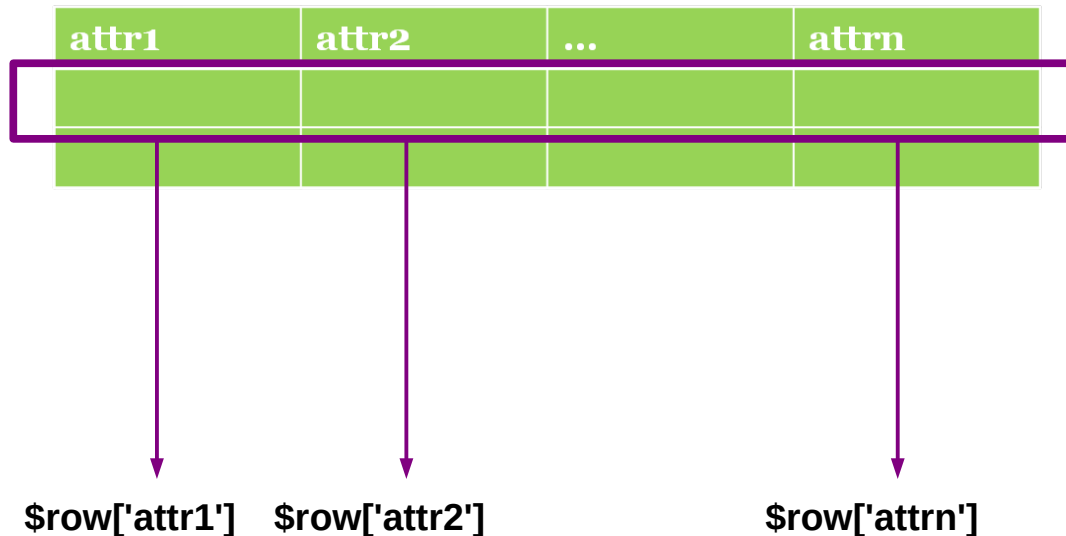
attr1	attr2	...	attrn

5. Ottenere il risultato

11

```
$row = mysqli_fetch_assoc($res);
```

```
$row = $res->fetch_assoc();
```



5. Ottenere il risultato

12

```
$row = mysqli_fetch_array($res,flag);
```

```
$row = $res->fetch_array(flag);
```

attr1	attr2	...	attrn

\$row[0]
oppure
\$row['attr1']

\$row[1]
oppure
\$row['attr2']

\$row[n-1]
oppure
\$row['attrn']

flag = MYSQLI_NUM oppure MYSQLI_ASSOC

```
$obj = mysqli_fetch_object($res);
```

```
$obj = $res->fetch_object();
```



\$obj->attr1
\$obj->attr2
...
\$obj->attrn

5. Ottenere il risultato

13

- Sono disponibili altre funzioni/proprietà che possono essere usate dopo l'esecuzione delle query per **capire se le cose sono andate a buon fine**
- **SELECT**
\$rowcount=**mysqli_num_rows**(\$res);
\$rowcount=**\$res->num_rows**;
- **INSERT, UPDATE, DELETE**
\$num = **mysqli_affected_rows**(\$con);
\$num = **\$con->affected_rows**;

6. Formattare il risultato

14

Si deve produrre il **markup HTML** che verrà restituito al client.
Ad esempio, nel caso della seguente query

```
$q = "SELECT * FROM user WHERE name='$name' AND surname='$surname';"
```

supponendo di avere **una sola tupla nel risultato**, possiamo scrivere

```
...  
// leggo i valori nella riga $row  
$name = htmlspecialchars($row["name"]);  
$surname = htmlspecialchars($row["surname"]);  
// preparo l'output  
$output = "<p>Welcome $name $surname, "  
$output .= "we have special offers for you</p>";  
// restituisco l'output al client  
echo $output  
...
```

6. Formattare il risultato

15

Nel caso di un risultato con più tuple, il **codice** che produce il markup HTML **va ripetuto (while, for) per ciascuna tupla**

```
while ($row = mysqli_fetch_assoc($res) ) {
```

```
// leggo i valori nella riga $row
```

```
$name=htmlspecialchars($row["name"]);
```

```
$email=htmlspecialchars($row["email"]);
```

Esiste anche la funzione htmlentities()
che “encodes a larger set of characters”

```
...
```

```
...
```

```
// restituisco l'output
```

```
echo "$name, $email, ...";
```

```
}
```

6. [Liberare la memoria]

16

```
mysqli_free_result($res);
```

```
$res->free();
```

TRUE in caso di successo, FALSE in caso di errore

6. Chiudere la connessione

17

```
mysqli_close($con);
```

```
$con->close();
```

3. Preparare la query: INSERT

18

In questo secondo esempio la query “mischia” codice SQL e dati in arrivo dal browser



```
$query = "INSERT INTO users  
(id_user,name,surname,email,password) VALUES  
(NULL,'$name','$surname','$email','$password')";
```

Anche in questo caso vedremo i prepared statement

3. Preparare la query: INSERT

19

- Dopo aver eseguito la query si può verificare il numero di record coinvolti nell'operazione

```
$num = mysqli_affected_rows($con);
```

```
$num = $con->affected_rows;
```

- Le query di UPDATE e DELETE sono del tutto analoghe

https://www.geeksforgeeks.org/php-mysqli_real_escape_string-function/

Occhio alla stringa di connessione al database, da non replicare :(

Per lavorare con il database

20

- Crea un utente per il database del tuo progetto web
- Progetta il database su carta e, solo in seguito, crea il database e le tabelle
- Ricorda di “pulire” l’input che deve essere memorizzato nelle tabelle del database
- Ricorda di cifrare le password
- Ricorda di controllare i valori restituiti dalle query prima di inoltrare il risultato al client

Imparare dagli errori

21

- PHP genera diverse tipologie di errori
 - **fatal error**: errori critici che interrompono l'esecuzione
 - **parse error**: errori di sintassi che interrompono l'esecuzione
 - **warning**: errori generati a runtime che non interrompono l'esecuzione
 - **notice**: simili ai warning, non interrompono l'esecuzione
- I fatal error sono quelli più gravi perché bloccano l'esecuzione dell'applicazione

Imparare dagli errori

22

- Invece di visualizzare a video gli errori PHP (cosa che non si deve **mai fare in produzione**) si possono scrivere in un file di log
- La funzione PHP **error_log()** permette di scrivere gli errori in un file di log del web server o specificato come parametro, e anche di inviarli a un indirizzo email

error_log(message, type, destination, headers);

message è il testo del messaggio

type è un numero intero che specifica dove sarà scritto il messaggio (0 = file di log del web server, 1 = email, **3 = file specificato nel parametro destination**)

<https://www.php.net/manual/en/function.error-log.php>