# Correzione compito Form Login

Ricordare che in PHP bisogna usare **===** per scrivere il simbolo di uguaglianza, così si evitano i buchi mega galattici di PHP stesso.

Per i Form, usare sempre il **metodo POST**, perché molto più sicuro e privato. Poi non bisogna mettere i dati utente nella Session, non è sicuramente un’idea tanto buona.

Si possono usare istruzioni del tipo:

$azione = $\_GET['action'] **??** '';

L’operatore ternario va dal $\_GET fino in fondo. Se ci fosse anche isset(), comprende anche quello, essendo la condizione (T/F).

che assegna il valore della chiave se non è NULL, altrimenti assegna una stringa vuota.

Il Request Method permette di controllare il metodo con cui vengono passati i dati; sempre mettere **POST** con dati sensibili, come utente e password soprattutto.

Con **exit** si interrompe l’esecuzione della pagina (nel nostro caso interrompe l’esecuzione del file PHP), attenzione a usarlo bene! Non permette di andare avanti ad eseguire il codice. Con **return** invece, come sappiamo già, si termina l’esecuzione del blocco e si va avanti.

# PHP e MySQL

Ora prendiamo la cartella “php\_mysql” (si trova nella cartella “htdocs” di XAMPP).

Vediamo il codice in “crea\_database.php”.

I parametri di connessione al server MySQL (MariaDB) vanno indicati separati per la libreria MySQL. La libreria già incapsulata in Apache è “mysqli”.

Ricordare che non bisogna più fare la **concatenazione** ma l’**interpolazione** delle stringhe, con le graffe, per estrarre i valori e unirli alla stringa. Ricordare che **echo** in teoria non vorrebbe le parentesi…

La modalità di accesso al Database è disconnessa (come nei linguaggi classici). Quando abbiamo i dati in memoria, posso chiudere la connessione al Database.

Posso interpretare i miei risultati come **array a indice numerico** (meno consigliato), usando **fetch\_array()**; come **array associativo**, con **fetch\_assoc()**; con **fetch\_object()** allora ottengo un **oggetto**, quindi devo usare la **sintassi freccia** tipo le struct in C/C++, quindi **->** .

Vediamo un po’ di file, creiamo il database e popoliamolo, successivamente visualizziamo i dati attraverso “richiesta.html”. Poi creiamo una select-option per creare una combobox con le marche di moto.

Bisogna connettersi al database e fare il popolamento dinamico ottenendo i dati con la query e poi ciclandoli bisogna popolare una select. Questa tecnica si chiama **popolamento dinamico**.

Non riusciamo a popolare la combobox direttamente dal database, non va bene in HTML, perché stiamo usando codice PHP, che non si può inserire in una pagina HTML.

Esercizio: fare un programma PHP in cui se viene passato un parametro (nome moto) vengono mostrati tutti i dati relativi, altrimenti niente. Creiamo una funzione leggiPiloti() che permetta di realizzare la consegna. Vedi il sorgente visualizza\_con\_select.php.

L’ideale sarebbe avere una libreria di funzioni nell’head della pagina, per ora. Poi scriveremo in altri file PHP le nostre funzioni e importeremo i file con **include**, che vedremo quando faremo la OOP in PHP. Nel nuovo Form che creiamo, il valore dell’attributo **Action** è la **nuova pagina** **PHP** che abbiamo creato. Come **metodo di richiesta** in questo caso usiamo **GET**. Poi dobbiamo popolare le options di questa select con le moto per permettere all’utente di scegliere. Ricordare che **se** **in un ciclo While voglio stampare** nella pagina **con echo**, devo separare il ciclo dal resto della stringa, oppure posso provare a concatenare/interpolare ma forse è un po’ incasinato. È comunque sempre meglio, perché più comodo ed elegante, interpolare le stringhe (con le graffe), niente puntino!

Poi dobbiamo mettere l’attributo **value** a ciascuna option, altrimenti non ha alcun valore associato!

Vedi i sorgenti creati da Victor e postati su Classroom.