PDO (PHP Data Object)

è una libreria per l'accesso a DB

E' agnostica rispetto al DB, cioè funziona su diversi DBMS (almeno in teoria)

Un dei vantaggi principali è che ripara "automaticamente" da SQL injection

.

CONNESSIONE AL DATABASE

Per connettersi alla libreria sono necessarie le stesse informazioni che abbiamo già visto

Stringa DSN (Data SourceName)

```
$dsn = 'mysql; host=localhost;dbname=db';
```

driver — IP o nome macchina — nome db

Le altre due informazioni, nome utente e password, vengono passate al costruttore

```
$pdo = new PDO($dsn,
                'utente', 'password');
```

Eventualmente si possono inizializzare alcune opzioni, come l'encoding dei caratteri, se dovesse servire, approfondite

GESTIONE DEGLI ERRORI

La libreria è pensata per lanciare eccezioni nel modo standard, se non gestite lo script viene terminato (cosa non buonissima, ma c'è di peggio) Più avanti le gestiremo, per il momento possiamo zittirle quando serve.

PHP INI FILE

In PHP si possono impostare i livelli di visibilità degli errori a livello dello script

← mostra di errori oppure no

```
ini_set('display_errors', 1);
ini_set('log_errors', 0);
```

] Impostazione durante lo sviluppo

↑

logga gli errori oppure no

Scambiando 0 e 1 abbiamo invece l'impostazione di produzione

QUERY COSTANTI

Il metodo per inviare una query costante al db è query, il cui parametro è appunto la query SQL

```
$stmt = $pdo->query('SELECT *
                     FROM tennisti');
```

Il valore di ritorno, $stmt, è l'equivalente di una tabella.

Se utilizziamo il metodo fetchAll lo possiamo "convertire" in un vettore di righe, ognuna delle quali è una mappa

```
$tabella = $stmt->fetchAll();
```

QUERY VARIABILI E PREPARED STATEMENTS

La soluzione ovvia, ma sbagliata, è quella di costruire la query "a mano"

```
$sql = 'SELECT * FROM tennisti WHERE id = '
        . $id;
```

Una soluzione decisamente migliore è di utilizzare i prepared statements.
Un prepared statement è un'istruzione SQL che contiene dei "segnaposto", che verranno sostituiti dai valori corretti al momento dell'esecuzione

```
$sql = 'SELECT * FROM tennisti WHERE id = :id';
$stmt = $pdo->prepare($sql);
$stmt->execute([
        ':id' => $id,
]);
```

QUERY CON ISTRUZIONI DML

Valgono le stesse osservazioni viste in precedenza, l'unica differenza riguarda il fatto che non vengono ritornati dati

INSERT

Esempio

```
$sql = 'INSERT INTO tennisti
        (nome, cognome) VALUES
        (:nome, :cognome)'
$stmt = $pdo->prepare($sql);
$stmt->execute([
        'nome' => $nome,
        'cognome' => $cognome,
]);
```

Il metodo rowCount() applicato a $stmt restituisce il numero di righe coinvolte nell'operazione

UPDATE

```
$sql = 'UPDATE tennisti
        SET cognome = :cognome
        WHERE id = :id';
$stmt = $pdo->prepare($sql);
$stmt->execute([
    'cognome' => $cognome,
    'id' => $id
]);
```

DELETE

```
$sql = 'DELETE FROM tennisti
        WHERE categoria < :categoria';
$stmt = $pdo->prepare($sql);
$stmt->execute(...);
$righe_cancellate = $stmt->rowCount();
```