PDO - Php Data Objects: Connessione PHP – SQL

```php
<?php
$dbuser = 'dbuser';
$password = 'pwd';

try {
    // Connect to the database
     $dbh = new PDO('pgsql:host=localhost;dbname=test', $dbuser, $password);

    // SQL query and results
    $sql = "SELECT * FROM testrcd";
    foreach($dbh->query($sql) as $row) {
        echo $row[0], ': ', $row[1], '<br>';
    }
    $dbh = null;
} catch (PDOException $e) {
    // Display an error message if there is a problem
     echo "Error!: ", $e->getMessage(), '<br>';
    die();
}
?>
```

Si immagini di lavorare su un progetto che prevede la creazione di un sito Web basato sull'interazione con un DBMS per l'accesso ai dati, il Database manager utilizzato potrebbe essere per esempio MySQL; a questo scopo lo sviluppatore potrà sfruttare le potenzialità messe a disposizione dalle funzioni native mysql_ based o mysqli_ based, ma cosa succederebbe se, per una qualsiasi esigenza, ci si ritrovasse di fronte alla necessità di trasferire il sito Web su un altro server che ha come DBMS di riferimento PostgreSQL? O magari potrebbe accadere l'esatto contrario; oppure ancora si deve passare ad un DBMS proprietario come Oracle.

La soluzione per non riscrivere tutti i sorgenti PHP?

PDO

PDO per PHP rappresenta la miglior tecnologia al momento a disposizione per accedere a database utilizzando PHP, in quanto incorpora i vantaggi dell'approccio Object Oriented, delle prestazioni e infine di genericità e astrazione nell'accesso ai dati. In questo breve articolo vediamo alcuni esempi per la connessione utilizzando il driver PostgreSQL.

PHP 5 mette a disposizione degli sviluppatori **PDO**, *PHP Data Objects*, una estensione molto potente per la connessione ai database utilizzando una interfaccia consistente, **universale** (cioe' pressochè indipendente dalla particolare implementazione di SQL) e leggera.

L'accesso ai singoli database avviene sulla base di un driver PDO specifico (ad esempio esiste il driver PDO per PostgreSQL, come quello per MySQL, etc.).

In gergo tecnico, PDO fornisce un livello di astrazione all'accesso ai dati. PDO accede al database utilizzando direttamente le funzioni native messe a disposizione dal driver specifico.

Al momento rappresenta sicuramente il miglior modo per accedere a un sistema di gestione di database con PHP.

Connessione PHP con PDO

Per connettersi con PHP tramite PDO al database di esempio, è necessario definire il Data Source Name (DSN), in pratica la vecchia connection string. La sintassi in PostgreSQL prevede un prefisso pgsql: seguito da coppie variabile/valore per i seguenti possibili elementi:

- host: nome dell'host, indirizzo IP oppure directory per la socket locale Unix;
- port: porta del server PostgreSQL (di default 5432);
- dbname: nome del database;
- user: nome dell'utente per la connesione al database;
- password: relativa password;

Il DSN per l'esempio tabella articoli risulta pertanto essere:

```
'pgsql:host=localhost;dbname=postgres', 'Fabrizio', '' // Senza password
```

Il primo vantaggio dell'utilizzo di PDO che si evince subito è l'approccio Object Oriented. In caso di errore, il costruttore della classe PDO lancia una eccezione PDOException.

Se si include il codice all'interno di blocchi try/catch (vedi immagine all'inizio)

si migliora la gestione degli errori.

Esempio semplificato (ma funzionante) di connessione PDO (con tabella articoli):

```
<?php
/** Esempio di connessione PDO, OK per PHP / tabella articoli/Dic 2016

Stringa:  'pgsql:host=localhost;dbname=DBNAME', 'USERNAME', 'PASSWORD');
Meglio mettere la stringa in file e includerlo con include
**/

$myPDO = new PDO('pgsql:host=localhost;dbname=postgres', 'Fabrizio', '');

$risultato = $myPDO->query("SELECT * FROM articoli");

/*** echo "<pre>";
            foreach ($risultato as $row)
                   print_r($row)----> Interessante, prova a decommentare
**/

    echo "<table border=1>";

    echo "<th>Codice</th><th>Descrizione</th><th>Prezzo</th><tr>\n";
    // Titoli
```

```
    foreach ($risultato as $row)
    {
          echo "<td>".$row['codice']."</td>";
          /// Naturalmente e' OK anche $row[0]
          echo "<td>".$row['descrizione']."</td>";
          echo "<td>".$row['prezzo']."</td><tr>\n";
    }

    echo "</table>";
    $myPDO=null;    /// E' sufficiente questo per liberare risorse
?>
```

Conclusione:

Essenzialmente è possibile dire che PDO fornisce un **data-access abstraction layer**, cioè un livello di astrazione per l'accesso ai dati; il suo utilizzo permetterà di creare applicazioni dotate della massima portabilità possibile, l'utilizzatore potrà scegliere il DBMS di riferimento sulla base delle proprie esigenze senza particolari costrizioni relative alla tipologia di accesso ai dati dell'applicazione.

DBMS supportati:

- PDO_DBLIB (FreeTDS / Microsoft SQL Server / Sybase)
- PDO_FIREBIRD (Firebird/Interbase 6)
- PDO_IBM (IBM DB2)
- PDO_INFORMIX (IBM Informix Dynamic Server)
- PDO_MYSQL (MySQL 3.x/4.x/5.x)
- PDO_OCI (Oracle Call Interface)
- PDO_ODBC (ODBC v3 (IBM DB2, unixODBC and win32 ODBC))
- PDO_PGSQL (PostgreSQL)
- PDO_SQLITE (SQLite 3 and SQLite 2)
- PDO_4D (4D)

Approfondimenti:

https://code.tutsplus.com/tutorials/why-you-should-be-using-phps-pdo-for-database-access--net-12059