PDO LIBRARY CONNECTION

Here's how you use PDO to establish a connection to a MySQL server:

```
new PDO("mysql:host=hostname;dbname=database", 'username','password')
```

If you are using variable parameters, be careful with the "" (don't use single quotes)

```
$pdo = new PDO("mysql:host=$servername;dbname=$dbname",$username, $password);
```

In any case, it takes three arguments:
1. a string specifying the type of database (mysql:), the hostname of the server (host=localhost;), and the name of the database (dbname=ies)
2. the MySQL username you want PHP to use
3. the MySQL password for that username

There may be a connection error. You should catch the exception using a try...catch statement:

```
try {
$pdo = new PDO('mysql:host=localhost;dbname=ies','dwes', '2DAWdwes');
echo 'Database connection established.';
}
catch (PDOException $e) {
echo 'Unable to connect to the database server.';
}
```

PDO: CONFIGURING THE CONNECTION

Our first task is to configure how our PDO object handles errors. By default, PDO switches to a "silent failure" mode after establishing a successful connection.

We'd like our PDO object to throw a PDOException any time it fails to do what we ask.We can configure it do to so by calling the PDO object's setAttribute method:

```
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
```

what we're saying with this line is that we want to set the PDO attribute that controls the error mode (PDO::ATTR_ERRMODE) to the mode that throws exceptions (PDO::ERRMODE_EXCEPTION)

By default, when PHP connects to MySQL, it uses the simpler ISO-8859-1 (or Latin-1) encoding instead of UTF-8.

To change it:

```
$pdo->exec('SET NAMES "utf8"');
```

PDO LIBRARY CONNECTION

Here's the complet code:

- including a status message that indicates when everything has gone right.

- including what caused the exception. Remember from Unit3, we can ask for the error message stored in the exception with getMessage() method

```
try
{
$pdo = new PDO('mysql:host=localhost;dbname=ies','dwes', '2DAWdwes');
$pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
$pdo->exec('SET NAMES "utf8"');
}
catch (PDOException $e)
{
echo 'Unable to connect to the database server: ' . $e->getMessage();
exit();
}

echo 'Database connection established.';
...
$pdo = null; // force to disconnect from the database server
```

PDO LIBRARY: SENDING SQL QUERIES

To execute the query:

For DELETE, INSERT, and UPDATE queries (which are used to modify stored data), the exec method returns the number of table rows (entries) that were affected by the query.

```
$affectedRows=$pdo->exec($sql)
```

With exec you can execute multiple instructions:

```
$sql = "
INSERT INTO car(name, type) VALUES ('car1', 'coupe');
INSERT INTO car(name, type) VALUES ('car2', 'coupe');
";

$db->exec($sql);
```

For SELECT queries, the query method returns a list of all the rows (entries) returned from the query.

```
$result=$pdo->query($sql)
```

PDO: UPDATE EXAMPLE - exec

Set the dates of all comments that contained the word "brilliant"...

```
try
{
// connection
...

$sql = 'UPDATE comment SET commentdate="2024-04-03" WHERE commenttext LIKE
"%brilliant%"';
$affectedRows = $pdo->exec($sql);
}

catch (PDOException $e)
{
echo 'Error performing update: ' . $e->getMessage();
exit();
}

echo "Updated $affectedRows rows.";
```

Insert and Delete statements work the same

PDO: SELECT EXAMPLE - query

The "query" method returns a PDOStatement object with a result set containing a list of all the rows (entries) returned from the query.

To process all the rows – while...loop

And "fetch" method of the PDOStatement object, returns the next row in the result set as an associative array. When the rows end it returns false

```
try {
$pdo = new PDO('mysql:host=localhost;dbname=ijdb;charset=utf8','dwes',
'2DAWdwes');
$pdo->setAttribute(PDO::ATTR_ERRMODE,PDO::ERRMODE_EXCEPTION);
$sql = 'SELECT commenttext FROM comment';
$result = $pdo->query($sql);
while ($row = $result->fetch())
{
echo $row['commmenttext'];
}
} catch (PDOException $e) {
echo 'Unable to connect to the database server: ' .$e->getMessage() . ' in
' . $e->getFile() . ':' . $e->getLine();
}
```

PDO SECURITY CONCERNS SQL INJECTION ATTACK

We have the same problem as with mysqli.
And the same solution: prepared statement. In previous versions it was used "get_magic_quotes_gpc()" function but it has been DEPRECATED as of PHP 7.4, and REMOVED as of PHP 8.0.

So, Instead of using this:

```
$sql = 'INSERT INTO comment SET commenttext ="' . $_POST['commenttext']
. '", commentdate =CURDATE()';
$pdo->exec($sql);
```

You should use prepared statement:

With variables:

```
$sql = 'INSERT INTO comment values (:commenttext,CURDATE())';
$stmt = $pdo->prepare($sql);

$stmt->bindValue(':commenttext', $_POST['commenttext']);
$stmt->execute();
```

Or more concise

```
$stmt->execute([':commenttext' => $_POST['commenttext']]);
```

PDO SECURITY CONCERNS SQL INJECTION ATTACK

Or with question mark

```
$sql = 'INSERT INTO comment values (?,?)';
$stmt = $db->prepare($sql);
$comment = "Good idea!";
$date = "CURDATE()";

$stmt->bindParam(1, $comment);
$stmt->bindParam(2, $date);
$stmt->execute();
```

Or more concise

```
$stmt->execute([$comment,$date]);
```

Note: Remember both INSERT syntax:

```
INSERT INTO comment SET commenttext = 'Good idea';
INSERT INTO comment values ('Good idea',CURDATE());
```

TRANSACTIONS

Database transactions ensure that a set of data changes will only be made permanent if every statement is successful.

PDO run "auto-commit" mode by default: It means that every query that you run has its own implicit transaction.

Example: Let's assume that we are creating a set of entries for a new employee, who has been assigned an ID number of 23. In addition to entering the basic data for that person, we also need to record their salary.

```
<?
try {
  $pdo->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
  $pdo->beginTransaction();
  $pdo->exec("insert into staff (id, first, last) values (23, 'Joe',
'Bloggs')");
  $pdo->exec("insert into salarychange (id, amount, changedate)
      values (23, 50000, NOW())");
  $pdo->commit();//save changes

} catch (Exception $e) {
  $pdo->rollBack(); //undo changes
  echo "Failed: " . $e->getMessage();
}
?>
```