Date 19/02/2019

PDO

Page No.
Date:

PHP Data Objects (PDO) is a lightweight system to access data from database with PHP. PDO has different drivers for different SQL database vendors. PDO provides a data-access abstraction layer, which means that, regardless of which database you are using, you use the same functions to issue queries and fetch data.

Thus PDO allows you to access a variety of different database and provides an easy way to query & retrieve results. PDO requires a minimum php version of 5.1.

```
$dbCon = new PDO('mysql:dbname=album;host=localhost;charset=utf8','root','');
$dbCon->setAttribute(PDO::ATTR_EMMULATE_PREPARES, false);
$dbCon->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);

$stmt = $dbCon->query('SELECT * FROM album');
```

OR

```
$stmt = $dbCon->prepare('SELECT * FROM album WHERE id = :id');
$stmt->execute(array('id' => 1));
```

in the form of associative array → FETCH_ASSOC
Associative array & numeric array → FETCH_BOTH

Page No.
Date:

```
$rows = $stmt → fetchAll(PDO:: FETCH_OBJ);
// we can use fetch instead of fetchAll. fetch always
// returns a single row of result if it is more than 1 row.
```

Example: 'fetchAll' result:

```
Array
(
    [0] => stdClass Object
    (
        [id] => 1
        [artist] => The Military Wives
        [title] => In My Dreams
    )
)
```

'fetch' result:

```
stdClass Object
(
    [id] => 1
    [artist] => The Military Wives
    [title] => In My Dreams
)
```

```
foreach($rows as $row){
    echo '<br>'.'Artist: '.$row→artist;
    echo '<br>'.'Title: '.$row→title;
}
```

Date
19/02/2017

Error Handling

Page No.
Date:

We do not have to handle with try catch right away. We can catch it anytime that is appropriate. It may make more sense to catch it at a higher level like outside of the function that calls the PDO stuff.

```
function getData($db)->query("SELECT * FROM album");
{
    $stmt = $db->query("SELECT * FROM album");
    return $stmt->fetchAll(PDO::FETCH_OBJ);
}
try{
    getData($db);
} catch(PDOException $ex){
    some_logging_function($ex->getMessage());
}
```

We can hide the dangerous error messages in production by turning display_errors off and just reading your error log.

Reference: wiki.hashphp.org/PDO_Tutorial_for_MySQL_Developers
php.net/manual/en/intro.pdo.php

Date 19/02/2017

Transactions

Page No.
Date

Here is an example of using transactions in PDO. Calling beginTransaction() turns off auto commit automatically.

```
try {
    $db->beginTransaction();
    $db->exec("SOMEQUERY");

    $stmt = $db->prepare("SOMEOTHERQUERY?");
    $stmt->execute(array($value));

    $stmt = $db->prepare("YETANOTHERQUERY??");
    $stmt->execute(array($value2, $value3));

    $db->commit();
} catch (PDOException $ex) {
    //something went wrong rollback!
    $db->rollBack();
    echo $ex->getMessage();
}
```

Date
21/02/2017

utf8mb4 instead of utf8 only

Specify the utf8mb4 character set on all tables and text columns in your database. This makes MySQL physically store & retrieve values encoded natively in UTF-8. Note that MySQL will implicitly use utf8mb4 encoding if a utf8mb4_* collation is specified (without any explicit character set).

On older versions of MySQL (< 5.5.3), we all unfortunately be forced to use simply utf8, which only supports a subset of Unicode characters. I wish I were kidding.