Important Notes:

Objects

* Data type that can store and manipulate values
* Class defines what an object can do
* Class is a collection of related functions and values like packaged in a single unit

Features of PDO

* PDO produces database neutral code which work with multiple database systems including mysql, mssql, postgres and sqlite.

Database Objects

* PDO and MySQLi objects connect to the database
* Use the objects methods to interact with the database
* Methods can take arguments
* Class constants are often used as arguments
* PDO::FETCH\_ASSOC, MYSQLI\_ASSOC
* Methods can return a new object

e.g. $result = $db->query($sql);

Notice such type of object no need to require declare new keyword.

* New objects will have the methods its own

e.g. $row = $result->fetch();

MySQL Improved (MySQLI)

* MySQLi is the dominant database used with PHP
* Has some MySQL features not supported by PDO
* MySQLi is compatible with Maria DB
* Original MySQL functions deprecated since PHP5.5
* Two interfaces: procedural and object-oriented
* No significant difference in performance
* Object-oriented interface is more concise
* Code is easier to read

Prepared Statements

* PDO & MySQLi supports prepared statements
* Prepared statements provide important security feature
* Template for SQL query that uses values from user input
* Placeholders for values stored in variables
* Prevents SQL injection because PDO and MySqli escapes quote and other characters before executing the query.
* More efficient when same query is reused
* Can bind results to named variables
* Example: $sql = ‘SELECT user\_id, first\_name, last\_name

FROM users

WHERE username = ? AND password = ?’;

* Here, ? is the anonymous placeholder
* $sql = ‘SELECT user\_id, first\_name, last\_name

FROM users

WHERE username = :username AND password = :pwd’;

* Here, :username named placeholder

Anonymous & Named Parameter

* In anonymous position is important where named parameter position is not required because named is used explicitly.

Placeholders

* Can be used only for column values
* Cannot be used for column names or operators
* Non-numeric values are automatically wrapped in quotes

Using Prepared Statements

* Prepare and validate the SQL with placeholders
* Bind values to the placeholders
* Execute the statement
* Bind output values to variables (optional)
* Fetch the results

Efficiency

* Single Query
* When query is submitted only once
* Prepared statement makes two round-trips to server
* First time validate and optimize the sql and second sent the values to the placeholder
* Valid non-prepared statement executes immediately because validation and optimization and also execution in a single operation and involve value directly in the sql because it takes single round to the sql server.
* User input still needs to be sanitized

Repeated Query

* When the same query need to more than once in same script
* Prepared statement analyzes and optimizes SQL once
* Values are sent separately
* Non-prepared statement analyzes SQL every time this can application slow down noticeably

Transaction

* Set of SQL queries executed as a unit
* Operation is committed only if all parts succeed
* Transaction can be rolled back if an error occurs
* Particularly useful for financial transfers
* Prevents rows being modified by another connection

MySQL and MariaDB

* Data must use InnoDB engine (XtraDB in MariaDB)
* Some commands, such as DROP, cannot be rolled back
* MyISAM tables don’t support transactions

Database Source Name (DSN)

* Identifies which database to connect to
* Prefix followed by colon identifies PDO driver
* Name/value pairs separated by semicolons
* DSN format depends on the driver

Example DSNs

* MySQL
* $dsn = ‘mysql:host=localhost;dbname=oophp ’;
* $dsn = ’mysql:host=localhost;port=3307;dbname=oophp ’;
* SQLite3
* $dsn = ‘sqlite:/path/to/oophp.db’;
* MS SQL Server
* $dsn = ‘sqlsrv:Server=localhost;Database=oophp ‘;

Fetching Results

PDO has four different methods for fetching results from the database query

* Fetch() – gets the next row from a result set is most light
* fetchAll() – Creates an array containing all rows

Fetch all the results once and store them as a multidimensional array.

By default return both associative and index array.

* fetchColumn() – gets a single column from the next row

Specify the column position as argument to return the specific column otherwise by default first column is returned.

* fetchObject() – gets the next row as an object

Summary of the fetches method

* fetchColumn will be used for list
* fetchAll will be used for complete data set as an array
* fetch is widely used which will return one row at a time

Which should I Use?

* Query()
* Is a select method or select queries
* Returns different values on the type of queries
* Returns the result set for SELECT queries
* Returns the SQL with INSERT, UPDATE, and DELETE
* Query() is useful it returns if query execution succeed otherwise returns false
* Exec()
* Non select queries
* Returns the number of rows affected
* Better for INSERT, UPDATE, and DELETE
* Never be used for select queries because it doesn’t return result set

Error Message in PDO

* Array ( [0] => 42S02 [1] => 1146 [2] => Table 'oophp.nams' doesn't exist )
* Here, first element is sqlstate
* Second element is error code
* Third element tells us what the problem is in plain English

Some Useful PDO Methods, Properties & Constants

* Exec() – Execute a non-select query which also return number of affected rows
* getMessage() – return an error information
* query() – for executing a query
* PDO has four different methods for fetching result from database – fetch(), fetchAll(), fetchColumn(), fetchObject()
* PDO::FETCH\_ASSOC returns an associative array
* PDO::FETCH\_NUM returns an index array
* errorInfo() - returns an array of error information about the last operation performed by this database handle.
* setAttribute() – sets an attribute to the database handle.

Disadvantages of quote()

* Not supported by all PDO drivers
* Dependent on the database’s character set
* Character set must be set at server level or in the DSN
* Often slower than prepared statement

Embed user input with a prepared statement in PDO

Binding Values

* Binding values to placeholders
* Use bindParam() or bindValue()
* Pass an array of values to the execute() method
* Output Values
* Use bindColumn()
* Named variables simplify display in a web parge

Using bindParam()

* Binds only a variable
* Value not evaluated until statement is executed
* “Fatal error: Cannot pass parameter 2 by reference”

Using bindValue()

* Binds a known value
* Useful for setting a field to NULL
* $stmt->bindValue(‘:id’ , NULL, PDO::PARAM\_NULL);
* Can also be used with a variable

Anonymous Placeholders

* Use question (?) mark in the placeholders
* In anonymous use number in values instead of named values is start from 1
* The real drawback of the anonymous placeholder is if we change the sql then reorder the position
* In named placeholder it doesn’t need

Execute Method

* PDO allows to pass the value directly by the execute instead of binding value with the bindParam() or bindValue() as an array.
* When we do this all values are treated as string
* We can create the array first or define it directly in execute method.
* Passing the array to the execute is the convenient to short hand
* But it has disadvantages all values pass to as strings
* That means we can’t insert null values
* In anonymous placeholder it also danger for getting wrong order in making changes in the sql
* However it is more useful if we passing single value to the execute method even in this we need to use array.

Difference between bindParam() and bindValue()

* With bindParam, you can only pass variables ; not values
* with bindValue, you can pass both (values, obviously, and variables)
* bindParam works only with variables because it allows parameters to be given as input/output, by "reference" (and a value is not a valid "reference" in PHP) : it is useful with drivers that (quoting the manual)

Example of binding Values

[With bindParam] Unlike PDOStatement::bindValue(), the variable is bound as a reference and will only be evaluated at the time that PDOStatement::execute() is called.

So, for example:

$sex = 'male';

$s = $dbh->prepare('SELECT name FROM students WHERE sex = :sex');

$s->bindParam(':sex', $sex); // use bindParam to bind the variable

$sex = 'female';

$s->execute(); // executed with WHERE sex = 'female'

or

$sex = 'male';

$s = $dbh->prepare('SELECT name FROM students WHERE sex = :sex');

$s->bindValue(':sex', $sex); // use bindValue to bind the variable's value

$sex = 'female';

$s->execute(); // executed with WHERE sex = 'male'

Documentation

* PDO Constants

<http://www.php.net/manual/en/pdo.constants.php>

* [www.php.net/manual/en/pdo.drivers.php](http://www.php.net/manual/en/pdo.drivers.php)
* Only the $dsn is database specific
* All other PDO code is database-neutral

Note

* We can binding value by using bindParam() or bindValue() and also by using execute() method where passing value directly by an array.
* To bind the input value use the bindParam() or bindValue and if we want to bind the output value use bindColumn() method.

PHP Exception Handling

* Exceptions are used to change the normal flow of a script if a specified error occurs.
* This is what normally happens when an exception is triggered:
* The current code state is saved
* The code execution will switch to a predefined (custom) exception handler function
* Depending on the situation, the handler may then resume the execution from the saved code state, terminate the script execution or continue the script from a different location in the code

Basic Use of Exceptions

* If an exception is not caught, a fatal error will be issued with an "Uncaught Exception" message.

Try, throw and catch

Proper exception code should include:

1. Try - A function using an exception should be in a "try" block. If the exception does not trigger, the code will continue as normal. However if the exception triggers, an exception is "thrown"
2. Throw - This is how you trigger an exception. Each "throw" must have at least one "catch"
3. Catch - A "catch" block retrieves an exception and creates an object containing the exception information
4. <?php  
   //create function with an exception  
   function checkNum($number) {  
     if($number>1) {  
       throw new Exception("Value must be 1 or below");  
     }  
     return true;  
   }  
     
   //trigger exception in a "try" block  
   try {  
     checkNum(2);  
     //If the exception is thrown, this text will not be shown  
     echo 'If you see this, the number is 1 or below';  
   }  
     
   //catch exception  
   catch(Exception $e) {  
     echo 'Message: ' .$e->getMessage();  
   }  
   ?>
5. The code above will get an error like this:
6. Message: Value must be 1 or below

**Example explained:**

The code above throws an exception and catches it:

1. The checkNum() function is created. It checks if a number is greater than 1. If it is, an exception is thrown
2. The checkNum() function is called in a "try" block
3. The exception within the checkNum() function is thrown
4. The "catch" block retrieves the exception and creates an object ($e) containing the exception information
5. The error message from the exception is echoed by calling $e->getMessage() from the exception object
6. PDO used ? Mark for anonymous place-holder and taken input from users.
7. PDO also support named placeholder like :username or :pwd.
8. Using prepared statement for single query makes two round-trips to server. First time validate and optimize the sql and second send values to the placeholder.
9. If MySQL server runs on default settings, you don't need to specify that Default MySQL port is 3306.
10. If the included file is one level higher than the web page with the inclusion directive you can use:

CODE

include('../header.php');

and if it's two levels higher:

CODE

include('../../header.php');

1. Each object in php is unique.

For example the blueprint of a Car has following properties like color, mileage etc. Each real time object (car) has following properties but different like a car has red color others is blue etc.

1. Method dictates what an object can do.