**https://github.com/indieteq/PHP-MySQL-PDO-Database-Class**

**PDO Database Class**

A database class for PHP-MySQL which uses the PDO extension.

If you have any questions go to : <http://indieteq.com/index/readmore/how-to-prevent-sql-injection-in-php>

**To use the class**

**1. Edit the database settings in the settings.ini.php**

**Note if PDO is loading slow change localhost to -> 127.0.0.1 !**

[SQL]

host = 127.0.0.1

user = root

password =

dbname = yourdatabase

**2. Require the class in your project**

<?php

require("Db.class.php");

**3. Create the instance**

<?php

// The instance

$db = new Db();

**4. Logs - Modify the read/write rights of the root folder**

Everytime an exception is thrown by the database class a log file gets created or modified. These logs are stored in the logs directory. Which means the database class needs write access for the logs folder. If the files are on a webserver you'll have to modify the rights of the root folder otherwise you'll get a "Permission denied" error.

The log file is a simple plain text file with the current date('year-month-day') as filename.

**Examples**

Below some examples of the basic functions of the database class. I've included a SQL dump so you can easily test the database class functions.

**The persons table**

| **id** | **firstname** | **lastname** | **sex** | **age** |
| --- | --- | --- | --- | --- |
| 1 | John | Doe | M | 19 |
| 2 | Bob | Black | M | 41 |
| 3 | Zoe | Chan | F | 20 |
| 4 | Kona | Khan | M | 14 |
| 5 | Kader | Khan | M | 56 |

**Fetching everything from the table**

<?php

// Fetch whole table

$persons = $db->query("SELECT \* FROM persons");

**Fetching with Bindings (ANTI-SQL-INJECTION):**

Binding parameters is the best way to prevent SQL injection. The class prepares your SQL query and binds the parameters afterwards.

There are three different ways to bind parameters.

<?php

// 1. Read friendly method

$db->bind("id","1");

$db->bind("firstname","John");

$person = $db->query("SELECT \* FROM Persons WHERE firstname = :firstname AND id = :id");

// 2. Bind more parameters

$db->bindMore(array("firstname"=>"John","id"=>"1"));

$person = $db->query("SELECT \* FROM Persons WHERE firstname = :firstname AND id = :id"));

// 3. Or just give the parameters to the method

$person = $db->query("SELECT \* FROM Persons WHERE firstname = :firstname",array("firstname"=>"John","id"=>"1"));

More about SQL injection prevention : <http://indieteq.com/index/readmore/how-to-prevent-sql-injection-in-php>

**Fetching Row:**

This method always returns only 1 row.

<?php

// Fetch a row

$ages = $db->row("SELECT \* FROM Persons WHERE id = :id", array("id"=>"1"));

**Result**

| **id** | **firstname** | **lastname** | **sex** | **age** |
| --- | --- | --- | --- | --- |
| 1 | John | Doe | M | 19 |

**Fetching Single Value:**

This method returns only one single value of a record.

<?php

// Fetch one single value

$db->bind("id","3");

$firstname = $db->single("SELECT firstname FROM Persons WHERE id = :id");

**Result**

| **firstname** |
| --- |
| Zoe |

**Fetching Column:**

<?php

// Fetch a column

$names = $db->column("SELECT Firstname FROM Persons");

**Result**

| **firstname** |
| --- |
| John |
| Bob |
| Zoe |
| Kona |
| Kader |

**Delete / Update / Insert**

When executing the delete, update, or insert statement by using the query method the affected rows will be returned.

<?php

// Delete

$delete = $db->query("DELETE FROM Persons WHERE Id = :id", array("id"=>"1"));

// Update

$update = $db->query("UPDATE Persons SET firstname = :f WHERE Id = :id", array("f"=>"Jan","id"=>"32"));

// Insert

$insert = $db->query("INSERT INTO Persons(Firstname,Age) VALUES(:f,:age)", array("f"=>"Vivek","age"=>"20"));

// Do something with the data

if($insert > 0 ) {

return 'Succesfully created a new person !';

}

**Method parameters**

Every method which executes a query has the optional parameter called bindings.

The *row* and the *query* method have a third optional parameter which is the fetch style. The default fetch style is*PDO::FETCH\_ASSOC* which returns an associative array.

Here an example :

<?php

// Fetch style as third parameter

$person\_num = $db->row("SELECT \* FROM Persons WHERE id = :id", array("id"=>"1"), PDO::FETCH\_NUM);

print\_r($person\_num);

// Array ( [0] => 1 [1] => Johny [2] => Doe [3] => M [4] => 19 )

More info about the PDO fetchstyle : <http://php.net/manual/en/pdostatement.fetch.php>

**EasyCRUD**

The easyCRUD is a class which you can use to easily execute basic SQL operations like(insert, update, select, delete) on your database. It uses the database class I've created to execute the SQL queries.

Actually it's just a little ORM class.

**How to use easyCRUD**

**1. First, create a new class. Then require the easyCRUD class.**

**2. Extend your class to the base class Crud and add the following fields to the class.**

**Example class :**

<?php

require\_once("easyCRUD.class.php");

class YourClass Extends Crud {

# The table you want to perform the database actions on

protected $table = 'persons';

# Primary Key of the table

protected $pk = 'id';

}

**EasyCRUD in action.**

**Creating a new person**

<?php

// First we"ll have create the instance of the class

$person = new person();

// Create new person

$person->Firstname = "Kona";

$person->Age = "20";

$person->Sex = "F";

$created = $person->Create();

// Or give the bindings to the constructor

$person = new person(array("Firstname"=>"Kona","age"=>"20","sex"=>"F"));

$created = $person->Create();

// SQL Equivalent

"INSERT INTO persons (Firstname,Age,Sex) VALUES ('Kona','20','F')"

**Deleting a person**

<?php

// Delete person

$person->Id = "17";

$deleted = $person->Delete();

// Shorthand method, give id as parameter

$deleted = $person->Delete(17);

// SQL Equivalent

"DELETE FROM persons WHERE Id = 17 LIMIT 1"

**Saving person's data**

<?php

// Update personal data

$person->Firstname = "John";

$person->Age = "20";

$person->Sex = "F";

$person->Id = "4";

// Returns affected rows

$saved = $person->Save();

// Or give the bindings to the constructor

$person = new person(array("Firstname"=>"John","age"=>"20","sex"=>"F","Id"=>"4"));

$saved = $person->Save();

// SQL Equivalent

"UPDATE persons SET Firstname = 'John',Age = 20, Sex = 'F' WHERE Id= 4"

**Finding a person**

<?php

// Find person

$person->Id = "1";

$person->Find();

echo $person->firstname;

// Johny

// Shorthand method, give id as parameter

$person->Find(1);

// SQL Equivalent

"SELECT \* FROM persons WHERE Id = 1"

**Getting all the persons**

<?php

// Finding all person

$persons = $person->all();

// SQL Equivalent

"SELECT \* FROM persons

**Copyright and license**

**Code released under Beerware**