Should I Use MySQLi or PDO?

If you need a short answer, it would be "Whatever you like".

Both MySQLi and PDO have their advantages:

PDO will work on 12 different database systems, whereas MySQLi will only work with MySQL databases.

So, if you have to switch your project to use another database, PDO makes the process easy. You only have to change the connection string and a few queries. With MySQLi, you will need to rewrite the entire code - queries included.

Both are object-oriented, but MySQLi also offers a procedural API.

Both support Prepared Statements. Prepared Statements protect from SQL injection, and are very important for web application security.

* Prepared statements are very useful against SQL injections, because parameter values, which are transmitted later using a different protocol, need not be correctly escaped. If the original statement template is not derived from external input, SQL injection cannot occur.

A great benefit of PDO is that it has an exception class to handle any problems that may occur in our database queries. If an exception is thrown within the try{ } block, the script stops executing and flows directly to the first catch(){ } block.

MySQL Examples in Both MySQLi and PDO Syntax

In this, and in the following chapters we demonstrate three ways of working with PHP and MySQL:

* MySQLi (object-oriented)
* MySQLi (procedural)
* PDO

<?php  
$servername = "localhost";  
$username = "root";  
$password = "";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=myDB", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    echo "Connected successfully";   
    }  
catch(PDOException $e)  
    {  
    echo "Connection failed: " . $e->getMessage();  
    }  
?>

Insert Data

<?php  
$servername = "localhost";  
$username = "username";  
$password = "password";  
$dbname = "myDBPDO";  
  
try {  
    $conn = new PDO("mysql:host=$servername;dbname=$dbname", $username, $password);  
    // set the PDO error mode to exception  
    $conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);  
    $sql = "INSERT INTO MyGuests (firstname, lastname, email)  
    VALUES ('John', 'Doe', 'john@example.com')";  
    // use exec() beca use no results are returned  
    $conn->exec($sql);  
    echo "New record created successfully";  
    }  
catch(PDOException $e)  
    {  
    echo $sql . "<br>" . $e->getMessage();  
    }

$conn = null;  
?>

fetchALL();

PDOStatement::fetchAll — Returns an array containing all of the result set rows or FALSE on failure.

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "test";

try {

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

// set the PDO error mode to exception

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

// prepare sql and bind parameters

$stmt = $conn->prepare("select \* from tbl\_test");

$result=$stmt->execute();

$result=$stmt->fetchAll();

print\_r($result);

echo "New records created successfully";

}

catch(PDOException $e)

{

echo "Error: " . $e->getMessage();

}

$conn = null;

?>

Prepared statement

<?php

$servername = "localhost";

$username = "root";

$password = "";

$dbname = "test";

try {

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

// set the PDO error mode to exception

$conn->setAttribute(PDO::ATTR\_ERRMODE, PDO::ERRMODE\_EXCEPTION);

// prepare sql and bind parameters

$stmt = $conn->prepare("INSERT INTO tbl\_test VALUES (?,?,?)");

$stmt->bindParam(1, $id);

$stmt->bindParam(2, $name);

$stmt->bindParam(3, $add);

$id=2;

$name="shyam kumar";

$add="hetauda";

$stmt->execute();

echo "New records created successfully";

}

catch(PDOException $e)

{

echo "Error: " . $e->getMessage();

}

$conn = null;

?>