PDO

PDO = PHP data object (a secure way to connect to databases/a data access layer)

In order to securely connect to a database, we use a PDO instance:

//CONNECT TO DB (usually in a try{}catch(){} block)

try {

    //host, database name, user, password

 $pdo = new PDO("mysql:host=localhost;dbname=expenses", "root", "123456");

}catch(PDOException $e){

      die($e->getMessage());

}

Next we prepare the statement to be executed:

//prepare the statement

$statement = $pdo->prepare('select \* from expenses\_record');

Execute the statement:

//execute the statement;

$statement->execute();

Fetch and display all the results with fetchAll():

var\_dump($statement->fetchAll());//displays all records from table (no format)

Fetch as object:

var\_dump($statement->fetchAll(PDO::FETCH\_OBJ));//displays all records from table

\*dsn = data source name ()

PDO fetching data with positional parameters and named parameters;

\*this separates the SQL instructions from actual data;

Positional Parameters (?)

$host = 'localhost';

$user = 'root';

$password = '123456';

$dbname = 'expenses';

//set DSN (data source name)

$dsn = 'mysql:host=' . $host . ';dbname=' .$dbname;

//create PDO instance

$pdo = new PDO($dsn, $user, $password);

//Prepared statements (prepare() and execute());

//User input

$table = "expenses\_record";

$description = "dinner ingredients";

//positional params (supported by mysqli)

$sql = "SELECT \* FROM $table WHERE description = ?";

$stmt = $pdo->prepare($sql);

$stmt->execute([$description]);

$results = $stmt->fetchAll(PDO::FETCH\_ASSOC);

var\_dump($results);

//note that $description is given as a positional parameter (the ?) and it is inserted as actual variable only when the statement is being executed -->this separates SQL instructions from actual variables/data

In the example above, the $description variable is not being passed to the program until the execution of the statement (see -> $stmt🡪execute([$description]); Until that line, the description field was a placeholder (the ***?*** sign);

Named parameters (:param);

//named parameters (:param)

//User input

$table = "expenses\_record";

$description = "dinner ingredients";

//positional params (supported by mysqli)

$sql = "SELECT \* FROM $table WHERE description = :description";

$stmt = $pdo->prepare($sql);

$stmt->execute(["description"=>$description]);

$results = $stmt->fetchAll(PDO::FETCH\_ASSOC);

var\_dump($results);

//note that $description is given as a named parameter (the string :description) and at the execution of statement, the actual variable is being passed to the program through an associative array;

In the example above, the $description variable is being replaced in the SQL statement by a dummy named parameter (the string ***:description***); When the statement is being executed, the named parameter is being replaced with the actual variable which is being passed through an associative array (in which the named param corresponds to the actual variable);