**OBJECT->ORINTED->PROGRAMMING (PDO)(AJAX)(JSON)**

**What’s inside OOP:**

* **class**: Template for object. **//class human {};**
* **property**: var inside class called property. **//public $name = ‘’;**
* **method**: function inside class called method. **//public function (){};**
* **object**: when called a class using new kw. **//$obj = new human;**
* **$this-> :** This kw uses to declare a property or method inside a class.

**Uses of Property:**

* **public $var = “data”;**// Declare A property inside class.
* **$this -> var; //** use property inside method.
* **This** declare this property inside class.

**Make a property Constant:**

* **const NAME = “Nazmul Huda”;**// Declare A constant inside class.
* **:: //** **Scope Resolution Operator**.
* **self::NAME //Call constant inside a class.**
* class::NAME // Call Constant outside of a class. No need object.

**Make a method or property Constant using static KW:**

* **public static function job() {}** // Declare A constant Function.
* **class::job();**// Call a method without creating object.
* **self::job(); //**Call static method class.
* **public static $var = “”** // Declare A constant property.
* **class::$var;**// Call a property without creating object.
* **Why Static:** Sometimes we’ve no need using \_\_construct, \_\_destruct;

**Extends a Class in OOP:**

* **class A{}** // Declare a class.
* **Class B extends A{}** // **Child** can access **Parent** method & Properties.
* **class B** // Can overwrite class a properties and methods.
* **$this-> name** // Use child property or method child priority first.
* **parent::job();** // Use parent:: to use parent class method in child.
* **$this-> name** // Use parent property in child class.

**Use Final KW in class & Method:**

* **final class A{} /**/ Can’t extend Class A with any child class.
* **public final function job(){}** // Can use But can’t overwrite final method.

**Abstract Class:**

* **abstract class abcd{}/**/ Can’t instantiate abstract class/Can’t create Object.

**Access Modifier In OOP:**

* **public //** Can be access from everywhere.
* **protected //** Can be access from own and extends class.
* **private //** Only Can be access from own.

**\_\_construct & \_\_destruct method in OOP:**

* **function \_\_construct(){} //** Load automatically at first of a program.
* **function \_\_destruct(){} //** Load automatically at last of a program.
* **$var = new //** load automatically when a class will instantiate.

**Sent Data By Object & Received by \_\_construct:**

* **$obj = new abed(‘name’, ‘age’, ‘job’); //** Pass Data when creating object.
* **\_\_construct($name, $age, $job) //** Received values by construct.
* **$this-> name = $name //** Set value to it’s properties.

**Connect Class By spl\_autoload\_resister():**

* **$obj = new Human(); //** When object will instantiate.
* **spl\_autoload\_resister(function($class){ require\_once “$class.php” }).**
* **{ require\_once “$class.php” } //** Connect with instantiate object.
* **Why this Function : To Upload Specific class when needed.**

**NameSpace: (Declaration should be the very first statement);**

* **namespace folder1\folder2; //** Set Namespace Before Class.
* **$obj = new folder1\folder2\class;//** Instantiate object using namespace.
* **use folder1\folder2\file\_name //** **Use KW** for **namespace**.
* **use folder1\folder2\class as abcd //** Also can change Class name.
* **Why: For Setting Root Folder.**

**namespace Using spl\_auto\_resister():**

* **namespace folder1\folder2\folder3; //** Set Namespace using backslash Class.
* **spl\_auto\_resister(function($class){** require\_once “$class”. “.php”; )}
* **use folder1\folder2\folder3\lass\_name //** To knew the root.
* **use mysqli; // When use pre build function inside another function.**
* **If(file\_exist($class\_name)){ require\_once “class”;};**
* **dirname(\_\_FILE\_\_). “autoload.php”. //Get the exact Directory name.**
* **Class: notice that Class folder is not allowed in class directory.**
* **Why**: Use namespace and spl\_auto\_resister() **for clean root folder.**

**.htaccess**

* **.htaccess** // Create A file with htaccess ext.
* **options – indexes // Access Forbidden indexes files.**
* **Here + (is allow), & - (is Disallow).**

**PDO(Php Data Object)**

**Mysqli:**

* new mysqli(‘host’, ‘user’, ‘pass’, ‘db’);
* $connection -> query($sql);
* Statement : query();

**PDO: (**Case Sensitive):

* new **PDO**(‘mysql:host=localhost; dbname= db, ‘username’, ‘pass’);//Mysql DB.
* new **PDO**(‘oci:host=localhost; dbname=db, ‘username’, ‘pass’); //Oracle DB.
* query() //Also allowed in pdo but standard is prepare();
* Statement : prepare(); //Must have to execute().
* Data Pass by execute(); for more security.
* For insert Values declare placeholder eg: ( :name, :email, :pass);

**Example DATA INSERT:**

$conn = new PDO('mysql:host=localhost; dbname=pdo\_test','root','');

$sql = "INSERT INTO admin (name, email, cell) VALUES (:name, :email, :cell)";

$statement = $conn -> prepare($sql);

$statement -> execute([

':name' => $name,

':email' => $email,

':cell' => $cell

]);

**Example DATA FETCH:**

$conn = new PDO('mysql:host=localhost; dbname=pdo\_test','root','');

$sql = "SELECT \* FROM admin";

$statement = $conn -> prepare($sql);

$statement -> execute();

$data = $statement -> fetch();//fetch(PDO::FETCH\_ASSOC//OBJ//BOTH);

$data = $statement -> fetchAll(); //Call Creates Multidimensional Array.

* While looping = when fetch();
* Foreach looping = when fetchAll();

**Ajax(Asynchronous JavaScript And Xml)**

* **$.ajax({});** // Call Ajax.
* **$.ajax({ url: ‘page.php’, success: function(abc){..get data..}});** **Go php file by url & get data by success method**.
* **$.ajax({ url: ‘page.php’, data: {name: let\_var, email: let\_var}, method: ‘POST’,** //**Send data by data method.**

**JSON(JavaScript Object Notation)**

* **Application & Server Contracting Language**.
* **{……} Json Object**. //Multiple Property can have in json object
* **[……..] Json Array.** //Multiple Object can have in json Array.
* **Structure: {'key’: value}// Keys must be in quotation if value is string be inside quotes.**
* **Use comma inside.**
* **Use “link”:[ ]; key to use an array inside object.**
* **Its totally An array of objects.**